Traffic flow on realistic road networks with adaptive traffic lights

Jan de Gier[1], **Timothy M Garoni**[2] **and Omar Rojas**[3,4]

[1] Department of Mathematics and Statistics, The University of Melbourne, Victoria 3010, Australia
[2] ARC Centre of Excellence for Mathematics and Statistics of Complex Systems, Department of Mathematics and Statistics, The University of Melbourne, Victoria 3010, Australia
[3] ARC Centre of Excellence for Mathematics and Statistics of Complex Systems, Department of Mathematics, La Trobe University, Victoria, 3086, Australia
[4] Department of Mathematics, University of Guadelajara, México

E-mail:
`jdgier@unimelb.edu.au`,`t.garoni@ms.unimelb.edu.au`,`orojas@up.edu.mx`

Abstract. We present a model of traffic flow on generic urban road networks based on cellular automata. We apply this model to an existing road network in the Australian city of Melbourne, using empirical data as input. For comparison, we also apply this model to a square-grid network using hypothetical input data. On both networks we compare the effects of non-adaptive vs adaptive traffic lights, in which instantaneous traffic state information feeds back into the traffic signal schedule. We observe that not only do adaptive traffic lights result in better averages of network observables, they also lead to significantly smaller fluctuations in these observables. We furthermore compare two different systems of adaptive traffic signals, one which is informed by the traffic state on both upstream and downstream links, and one which is informed by upstream links only. We find that, in general, both the mean and the fluctuation of the travel time are smallest when using the joint upstream-downstream control strategy.

PACS numbers: 05.60.Cd, 05.70.Ln, 89.40.Bb,89.75.Fb

AMS classification scheme numbers: 82C05,82C20,82C70,90B20

Submitted to: *Journal of Statistical Mechanics: Theory and Experiment*

1. Introduction

The study of vehicular traffic has played an increasingly significant role in non-equilibrium statistical mechanics over recent years. There are a number of approaches which may be taken to traffic modeling; see for example the reviews [1, 2, 3, 4, 5, 6]. The use of cellular automata (CA) has been the subject of much study within the statistical mechanics community ever since the seminal work of Nagel and Schreckenberg [7]. A cellular automaton is a model which is discrete in time, space, and state variables, whose dynamical rules are local. The Nagel-Schreckenberg (NaSch) model is generally considered to be the minimal model for traffic on freeways. A huge literature dealing with various extensions of the NaSch model has evolved since its first appearance, and our understanding of freeway traffic has benefited greatly as a result. It is safe to say however that the behavior of traffic *networks* is still far less well understood. Much of the progress on traffic networks made within the statistical mechanics community has been focused on regular lattices (see e.g. [8, 9, 10, 11, 12]), although there are some notable exceptions [13, 14, 15, 16].

The aim of the current work is to improve our understanding of traffic networks by studying a crucial aspect of such networks: traffic lights. The model we use for network traffic flow in this paper is CA based and applicable to arbitrary road networks. The optimization of traffic lights is a major challenge in urban traffic networks [17]. A natural idea, studied by several authors, is to link together or synchronize traffic lights [18, 11, 19, 10, 20, 21]. It is acknowledged however that more flexible strategies are needed than just fixed-cycle controls [22, 23, 24, 25]. In this paper we will study an adaptive control strategy, for which traffic light switching schedules may be acyclic and green time durations are not fixed. Green times are determined by instantaneous local traffic information, such as up- and downstream vehicle densities. The main ideas of this approach have been discussed in [26].

Specifically, we apply our CA model to an actual urban road network in the Melbourne suburb of Kew, using empirical data as input, and then study the effect of applying different types of traffic signal systems. We also study our CA on a square-lattice network in order to test robustness and network independent features. The development of a realistic and computationally efficient network traffic model based on an existing system of traffic lights, is part of an ongoing collaboration with the Roads Corporation of Victoria (VicRoads) in Australia.

1.1. Adaptive traffic signal systems

The rules governing the traffic signals at signalized intersections in urban road networks play a crucial role in determining the network's overall efficiency. A number of *adaptive* traffic signal systems exist and are in use around the world. The *Sydney Coordinated Adaptive Traffic System* (SCATS) is a traffic signal system used to control traffic lights in numerous cities around the world, including Sydney, Melbourne, Shanghai and Detroit. SCATS is adaptive in the sense that it uses knowledge of the recent state of traffic in the network to choose appropriate values of the parameters controlling the traffic lights, such as the amount of green time given to the various possible movements through each signalized intersection. However, the only input data to which SCATS has access is the data provided from existing induction-loop detectors, usually located at the stop line, and this information is rather limited. The use of more sophisticated detectors on our roads, allowing the collection of more detailed data such

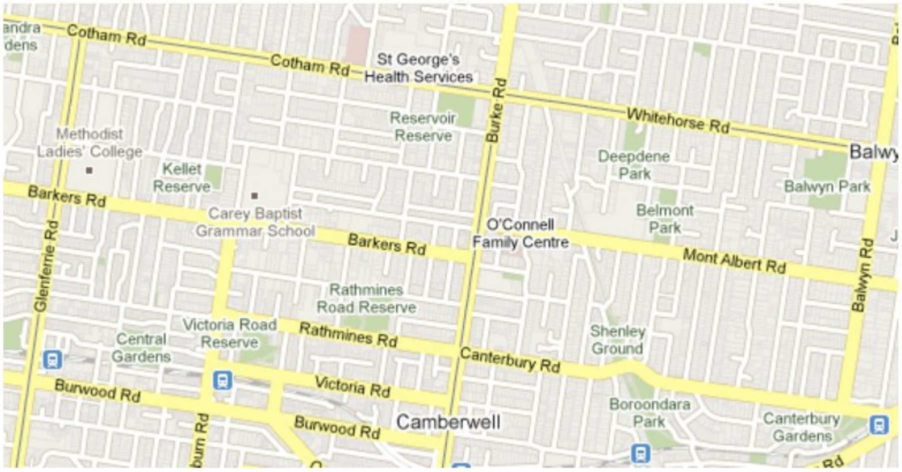


Figure 1. Google map of the chosen network (main roads only) in Kew, Melbourne, Australia. The size of the network is approximately 3.8km by 1.7km.

as instantaneous link densities and queue lengths, has the potential to significantly increase the efficiency of urban road networks. It is therefore of significant interest to investigate generalized adaptive traffic signal systems, which utilize more detailed input data, such as the density of incoming and/or outgoing links, and the most practical way to do that is via numerical simulation. By studying such generalized adaptive schemes we may hope to gain insight into the potential benefits of installing more sophisticated detectors on our roads.

Recently, certain types of adaptive or "self-organizing" traffic lights (SOTL) have been receiving attention in the statistical physics literature [17, 27, 26, 28, 29]. Self-organizing traffic lights have been investigated in the simple context of a Manhattan-like network in [26]. In such a network each intersection has only two possible signal *phases*‡; either eastbound traffic has a green light and northbound traffic has a red light, or vice versa. We have generalized the ideas presented in [26] to handle intersections with multiple signal phases. This generalization from two to multiple phases allows a much richer variety of behavior. With only two signal phases, the only question one can consider is "how long should the active phase run before switching to the other phase." With more than two phases however, the more interesting question of "which phase should we switch to next" also arises.

A further significant generalization that we introduce is that we not only consider the state of the *upstream* links which feed into a given intersection, but also the *downstream* links which are fed by the intersection. The idea being that not only is it important to give green time to a movement that will allow a congested upstream link to dissipate, but also that it is counterproductive to give green time to a movement that will further congest an already over-saturated downstream link. We find that for the Kew network, with boundary conditions corresponding to morning peak hour, the upstream-downstream adaptive traffic lights are approximately 5% more efficient

‡ Clearly this usage of the word "phase" is unrelated to the usual meaning in statistical mechanics. Its widespread use in the traffic engineering literature hopefully sanctions our use of it here.

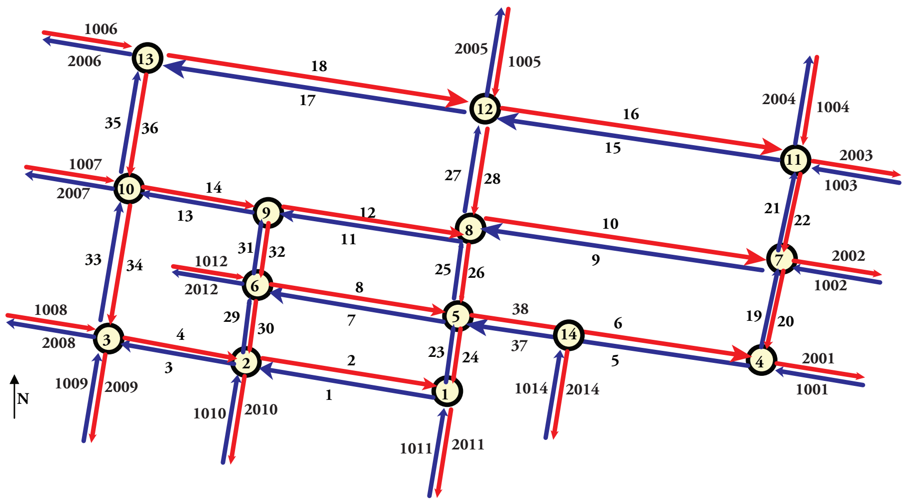


Figure 2. The network studied in our simulations, corresponding to the actual network in figure 1. Links with both endpoints shown are *bulk links*, while links with only one endpoint shown are *boundary links*. All link and node labels are arbitrary, but are used for reference within the text.

than the simple upstream-only version.

In order to test the robustness of these results, we then repeated the simulations on a square-lattice network, under a variety of boundary conditions. Specifically, we studied three choices of boundary conditions; strong westbound bias, uniform high density, and uniform low density. In the first two cases, our simulations again suggested that the upstream-downstream adaptive traffic lights performed better, being approximately 2-5% more efficient. For the low-density network, by contrast, there was no discernible difference between the two.

2. A cellular automata model for generic urban road networks

For clarity of presentation, we first sketch the main features of our traffic network model. A detailed algorithmic description is deferred to Appendix A.

We represent a road network by a directed graph, composed of *nodes* (i.e. intersections) and *links* (ordered pairs of nodes, i.e. streets); see figure 2. With each link is associated an ordered list of lanes, and each lane is a simple one-dimensional CA obeying Nagel-Schreckenberg [7] dynamics. Arbitrary street lengths are implemented in the model by allowing the lanes on each link to have an arbitrary number of cells.

The speed v of each vehicle can take one of $v_{\max} + 1$ allowed integer values $v = 0, 1, 2, \ldots, v_{\max}$. Taking the length of a cell to be $7.5m$ (corresponding to the typical space occupied by each vehicle in a jam) and the duration of each time step to be 1 second suggests $v_{\max} = 3$ is a reasonable choice for an urban network; i.e. each vehicle can move $0, 1, 2$ or 3 cells per time step in such a CA model, depending

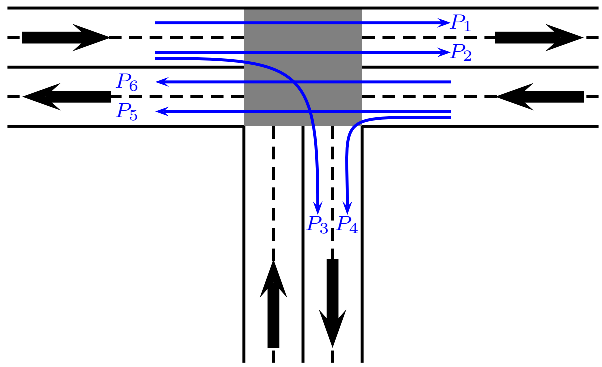


Figure 3. Typical example of a node in a road network. Here there are three inlinks and three outlinks, each consisting of two lanes. Each path P_i is an ordered pair $(in-lane, out-lane)$, and $\mathcal{P} = \{P_1, P_2, P_4, P_5, P_6\}$ is a typical example of a phase associated with this node, consisting of five paths.

on local traffic conditions. At each time step the positions of all vehicles are updated simultaneously, or *in parallel*, so that each vehicle makes its decision based on the *same* information.

Lanes can act as turning lanes by inserting *obstacles* into an appropriate number of cells at the beginning of the lane. In addition, neighboring lanes on a given link can interact via lane changing, the details of which we discuss in Appendix A.2. Thus, the dynamics along each given link is essentially a standard CA freeway model [1], albeit with input and output rates that are determined dynamically by the rest of the network. The complication arises in how to glue these one-dimensional CA together to form a network. We have chosen the following simple model.

2.1. Paths and Phases.

We define a *path*§ on node n to be an ordered pair $(\lambda_{\text{in}}, \lambda_{\text{out}})$, where λ_{in} (λ_{out}) is a lane directed to (from) n. We implement the topology of a road network by assigning to each node a list of *paths*. When a vehicle reaches the end of a link it can only move to another link along one of the node's paths; see figure 3. If $P = (\lambda_{\text{in}}, \lambda_{\text{out}})$ then we shall write $\text{in}(P) = \lambda_{\text{in}}$ and $\text{out}(P) = \lambda_{\text{out}}$. At a given node, we define a *phase* to be a particular subset of that node's paths. With each node is associated a set of phases $\mathcal{P}$, and at any instant precisely one phase is declared to be the *active phase* for that given node. The effect of traffic lights is then implemented by demanding that vehicles may only traverse a path if it belongs to the active phase. The dynamics for how the active phase is chosen at each node, at each instant of time, is a crucial aspect of the network's dynamics, and can change the network's efficiency dramatically; see section 3.

Within a given phase, we also allow each path to have a list of other paths to which it must give way. This allows us to model the fact that right turning vehicles often must give way to oncoming traffic‖, even though they have a green light. For

§ If one considers the road network as a directed multigraph, with lanes as edges, our paths are genuine graph-theoretical paths, of length 2.

‖ Assuming vehicles drive on the left side of the road.

example, if we added the path P_3 to the phase $\mathcal{P}$ in figure 3 then path P_3 would need to give way to paths P_5 and P_6. If at a given time-step there would happen to be vehicles wanting to traverse both P_3 and P_6 for instance, then only the vehicle wanting to traverse P_6 could proceed, while the vehicle wanting to traverse P_3 would stop at the end of P_3's inlink.

2.2. Turning probabilities.

In order to mimic origin-destination behavior, we demand that each vehicle makes a random decision about which link it wants to turn into at the approaching intersection. More precisely, for each node n, we assign to each ordered pair (l, l'), where l is an inlink and l' an outlink of n, the probability $\mathbb{P}(l \to l')$, that a vehicle on l wants to turn into l' when it reaches n. In our model, the turning decision is made when the vehicle first enters l, since its choice of which link to turn into at the approaching intersection should influence its dynamics as it travels along l. In particular it influences the vehicle's choice of when to change lanes; see Appendix A.2.

2.3. Boundary conditions.

When simulating a network, we must decide precisely where to put the *boundary*. Since all road networks are necessarily open systems, it must be the case that in any chosen network, some of the nodes are connected to links whose other endpoint is *not* part of the network. Any link with both endpoints contained in the chosen network we call a *bulk link*, whereas any link with precisely one node in the chosen network we call a *boundary link*. We can further classify the boundary links as being either *boundary inlinks*, if their to-node belongs to the network, or *boundary outlinks*, if their from-node belongs to the network. In figure 2 for example, the bulk links have labels 1 through 38, while all the links 1001 through 1014 are boundary inlinks, and all the links 2001 through 2014 are boundary outlinks. Traffic flows into the network via boundary inlinks, and flows out of the network via boundary outlinks.

A natural question to ask is: "To what extent should we simulate traffic on the boundary lanes?" To resolve this question we need to consider what boundary data is required, and available. Most importantly, we need to have appropriate boundary data determining the inflow and outflow of vehicles from the simulated network. In addition, if the traffic signals are being operated by SOTL we need to input appropriate values for the chosen demand function for each boundary lane; we discuss this further in sections 3, 5 and 6.

Consider a boundary in-lane λ of length L_{physical} metres, and suppose we visualize a discretization of λ into $L = L_{\text{physical}}/7.5$ equally sized cells, as we would do when simulating λ using cellular automata. Unlike bulk lanes, it is not necessary to model boundary lanes in their entirety; we are free to model as much of λ as we find convenient. Let i denote the first cell of λ which is included in the CA. The two extreme cases are obviously $i = 1$ and $i = L$, however all choices in between are *a priori* sensible. See figure 4.

Regardless of our choice, for boundary in-lanes λ we are required to insert new vehicles into cell i with some given probability α_λ. It therefore makes sense to choose the amount of the boundary lane which we simulate using CA in such a way that α_λ is *easily* obtained empirically. For our simulations of the Kew network, the most readily available, and probably most accurate, source of data comes directly from the

Figure 4. Boundary in-lane λ for which CA modeling begins on cell i.

occupancies of the stop-line loop detectors used by SCATS in Melbourne ¶. Let $\rho_{\lambda,i}$ denote the density of position i in lane λ. If o_λ denotes the empirically measured (by SCATS) stop-line occupancy of lane λ, then $o_\lambda \approx \rho_{\lambda,L}$. This suggests that for our simulations of Kew it is most sensible to only model the very end of each boundary in-lane, since we then have $\alpha_\lambda \approx \rho_{\lambda,L}$, and therefore to a good approximation $\alpha_\lambda \approx o_\lambda$. The simplest such strategy is to simply model the last cell of λ, which, at each time step, is occupied with probability o_λ. This is the strategy we used for the simulations of the Kew network. We used a slightly different approach for the square-lattice simulations, as no boundary input data is available; see section 6. The input strategy is described in more detail in Appendix A.1.

By contrast, for boundary out-lanes, λ, we simply assume that $\rho_{\lambda,1} = 0$, which should be a reasonable assumption except in the case of total grid-lock. We then simply let any vehicle which wants to enter lane λ do so with probability 1, provided it has a green light.

Finally, we emphasize that these questions of the optimal way in which to choose the boundary is a generic problem encountered by all network simulations; it is not related to the particular method (such as cellular automata) chosen to simulate traffic inside the network.

2.4. Time-inhomogeneous boundary conditions.

The above discussion referred to the boundary conditions input into the model at a given instant of time. In general however, we may want to allow the boundary conditions to evolve as the simulation proceeds. With such boundary conditions we can, for example, study build-up and decay before and after the AM peak hour.

For each lane λ of each boundary link l we provide as input an M-vector

$$(\alpha_\lambda^{(1)}, \alpha_\lambda^{(2)}, \ldots, \alpha_\lambda^{(M)}).$$

At times $t = (j-1)\,T_B + 1, \ldots, j\,T_B$ we perform boundary inflow using the probability $\alpha_\lambda^{(j)}$, for each $j = 1, 2, \ldots, M$. After iteration $t = M\,T_B$ the simulation terminates. The system therefore defines a non-stationary stochastic process. Consequently no stationary distribution can be assumed to exist, and there is no reason to assume time averages converge to anything meaningful (as would be implied by the ergodic theorem).

2.5. Overview of the model.

A high-level description of our CA model is presented in algorithm 1. The loop in algorithm 1 is over discrete time-steps, each time-step corresponding to 1 second. We emphasize that the dynamics defined by algorithm 1 correspond to updating all cells

¶ Data from 2009 for the Melburnian suburb of Kew has been made available to us by VicRoads

in *parallel*; i.e. all cells are updated based on the configuration at the *same* time step. In Appendix A we elaborate in detail on each step in algorithm 1.

Algorithm 1

loop
Inflow of vehicles into the network
Lane changes on each link
Mark the paths having vehicles wanting to traverse them
Nagel-Schreckenberg dynamics on each lane
Clear the marked paths on each node
Update active phase of each node
end loop

---

3. Self-organizing traffic lights (SOTL)

Suppose we agree on a suitable *demand* function $d(\mathcal{P})$ which quantifies the demand of each phase $\mathcal{P}$, of each given node. Phases with large values of $d(\mathcal{P})$ should be candidates for being the next choice of the active phase, $\mathcal{P}_{\text{active}}$. However, we must also keep track of the time $\tau(\mathcal{P})$ each phase has been idle, since we do not want a given phase to remain idle for too long, unless it has strictly zero demand. The key idea behind SOTL is to compute a *threshold* function, $\kappa(\mathcal{P})$, for each phase $\mathcal{P}$, which depends on both the phase's idle time and demand function, and when $\kappa(\mathcal{P})$ reaches a predetermined threshold value,

$$\kappa(\mathcal{P}) > \theta, \tag{1}$$

we consider making $\mathcal{P}$ the active phase. Perhaps the simplest reasonable quantity to use for the SOTL threshold function is

$$\kappa(\mathcal{P}) = d(\mathcal{P})\,\tau(\mathcal{P}). \tag{2}$$

Notice that $\kappa(\mathcal{P})$ is precisely zero whenever $\mathcal{P}$ has strictly zero demand, regardless of the size of $\tau(\mathcal{P})$. However, if $d(\mathcal{P}) > 0$ then $\kappa(\mathcal{P})$ grows monotonically with $\tau(\mathcal{P})$, so that $\kappa(\mathcal{P})$ will eventually become large even if $d(\mathcal{P})$ is small. This ensures that no driver can be left facing a red light indefinitely.

There are potentially an infinite number of sensible choices for κ and d that one could investigate. Regardless of the specific choices however, there should be a fixed cost associated with physically changing phases. To ensure we do not suffer excessively rapid switching between phases, we let $\tau(n)$ denote the amount of time node n has been in phase $\mathcal{P}_{\text{active}}$ and only allow n to change its phase if $\tau(n) \geq T_{\min}$, for some fixed parameter $T_{\min}$. We use $T_{\min} = 5$ throughout this paper.

Algorithm 2 presents a general SOTL protocol for governing the signals on a node n with phases $\Pi = \{\mathcal{P}_1, \mathcal{P}_2, \ldots\}$ (UAR abbreviates *uniformly at random*). When $\tau(n)$ becomes larger than $T_{\min}$, the algorithm determines the set of phases Π' for which the threshold function κ exceeds the threshold value θ. From Π' it then chooses the phases which attain the largest value of the threshold function, and among those it selects the phases which have been idle longest. Out of this latter set, called Π''', a phase is chosen at random to be the next active phase. In practice we find that Π''' almost always contains not more than one element.

Algorithm 2 (Acyclic SOTL)

Increment $\tau(n)$
for *each phase* $\mathcal{P} \neq \mathcal{P}_{\text{active}}$ ***do***
  Increment $\tau(\mathcal{P})$
end for
if $\tau(n) \geq T_{\min}$ ***then***
  Let $\Pi' = \{\mathcal{P} \in \Pi : \kappa(\mathcal{P}) > \theta\}$
  if $\Pi' \neq \emptyset$ ***then***
    Let $\Pi'' = \{\mathcal{P} \in \Pi' : \kappa(\mathcal{P}) = \max_{\mathcal{P}' \in \Pi'} \kappa(\mathcal{P}')\}$
    Let $\Pi''' = \{\mathcal{P} \in \Pi'' : \tau(\mathcal{P}) = \max_{\mathcal{P}' \in \Pi''} \tau(\mathcal{P}')\}$
    UAR, choose $\mathcal{P} \in \Pi'''$ *and set* $\mathcal{P}_{\text{active}} = \mathcal{P}$
    Set $\tau(\mathcal{P}_{\text{active}}) = 0$
    Set $\tau(n) = 0$
  end if
end if

This implementation of SOTL is acyclic in the sense that we do not impose any fixed ordering on the phases. One could also easily define a cyclic version of SOTL which uses a threshold function only to determine *when* to switch to the next phase in the fixed cycle.

Now let us consider the phase demand function in more detail. Given a suitable demand function $d(P)$ defined on paths P, we define the demand of the phase $\mathcal{P}$ as

$$d(\mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{P \in \mathcal{P}} \frac{d(P)}{\sigma_P}. \tag{3}$$

Thus, the demand of the phase is just a weighted sum of the demands of each path it includes. Let us now comment on the weighting. The parameter σ_P denotes the number of paths belonging to node n which have in-lane $\text{in}(P)$. For example, for the node in figure 3 we have $\sigma_{P_1} = \sigma_{P_6} = 1$ and $\sigma_{P_i} = 2$ for $i = 2, 3, 4, 5$. We refer to σ_P as the *degeneracy* of P. The weight $1/\sigma_P$ is included simply to ensure that paths with different degeneracies are weighted fairly, relative to each other. For example, consider again the node in figure 3. The demand of the phase $\mathcal{P} = \{P_1, P_2, P_4, P_5, P_6\}$ is

$$d(\mathcal{P}) = \frac{1}{5}\left(d(P_1) + \frac{d(P_2)}{2} + \left(\frac{d(P_4)}{2} + \frac{d(P_5)}{2}\right) + d(P_6)\right). \tag{4}$$

The phase $\mathcal{P}$ services all the possible paths emanating from the lanes $\lambda_1 = \text{in}(P_1)$, $\lambda_2 = \text{in}(P_6)$ and $\lambda_3 = \text{in}(P_4) = \text{in}(P_5)$, while it only services one of two possible paths, namely P_2, emanating from $\lambda_4 = \text{in}(P_2) = \text{in}(P_3)$. Therefore, the contributions to $d(\mathcal{P})$ corresponding to $\lambda_1, \lambda_2, \lambda_3$ all occur with unit weight, while that for λ_4 occurs only with weight $1/2$. Note that weighting by $1/\sigma_P$ implies that we are implicitly assuming all paths with the same in-lane are equally important; one could choose other weightings if, for a given in-lane, there were a reason to prefer one path over another.

Now let us move from the general to the concrete, and introduce the following two-parameter family of path demand functions

$$d(P) = \rho_{\text{in}(P)}^m (1 - \rho_{\text{out}(P)})^n, \tag{5}$$

where $\rho_{\mathrm{in}(P)}$ and $\rho_{\mathrm{out}(P)}$ are the instantaneous space averaged densities on the lanes in(P) and out(P). The factor $\rho_{\mathrm{in}(P)}^m$ in (5) implies that it is desirable to give a green light to paths which have a congested in-lane. This is intuitively reasonable, and in fact similar schemes are already applied in practice by systems such as SCATS (although not by quantifying congestion in terms of the actual lane density). The factor $(1 - \rho_{\mathrm{out}}(P))^n$ has a complementary effect – it provides a disincentive to giving a green light to a path whose out-lane is already congested. This is again intuitively reasonable, however it seems that this second mechanism has been far less widely applied in actual adaptive systems used in practice. The simulations presented in sections 5 and 6 were performed using SOTL with demand function (5), with both $(m, n) = (1, 0)$ and $(m, n) = (1, 1)$.

4. Observables

4.1. Density, speed, flow and queue length

We define the density, $\rho_l(t)$, of link l at time t to be the fraction of all cells on l which are occupied at that instant, and we define the space-mean speed, $v_l(t)$, to be the arithmetic mean of the speeds of all vehicles on link l. In general, l will contain multiple lanes, and we compute $\rho_l(t)$ and $v_l(t)$ by summing over all the cells/vehicles on all the lanes of l.

The flow, $J_\lambda(t)$, of lane λ during the tth time-step is simply the indicator for the event that a vehicle crosses the boundary between a fixed pair of neighboring cells during the tth update. The flow $J_l(t)$ on link l at time t is then simply the arithmetic mean of the $J_\lambda(t)$ over all $\lambda \in l$.

There is some ambiguity in deciding on an appropriate definition of exactly when a vehicle should be considered *queued*. We use the following simple prescription.

(i) When a vehicle first enters a link it is *un-queued*

(ii) A vehicle becomes *queued* if and only if:
   (a) It is stopped
   (b) Every cell in front of it is occupied

(iii) A queued vehicle remains queued until it turns into a new link

Some comments are in order. Firstly, in this definition, we insist that a vehicle must be stopped in order to be queued. This is reasonable, since a vehicle with speed 1 cell per iteration is traveling at around $27km/h$, and if a vehicle's speed is always at least $27km/h$ it does not seem sensible to say it was ever queued. Secondly, insisting on having all cells in front of the vehicle occupied is perhaps a little conservative, but it is certainly the simplest choice, and any other choice would be decidedly *ad hoc*. Finally, the rule that a queued vehicle only becomes dequeued when it leaves the current link is designed to take into account stop-and-go behavior. I.e. a vehicle that was stopped, and then starts again, only to stop again later never really left the queue. Armed with the above definition, we can now unambiguously define $Q_l(t)$ to be the number of vehicles on link l which are queued at time t.

Finally, we can compute network-mean observables $\rho(t)$, $v(t)$, $J(t)$, $Q(t)$, defined as the arithmetic means over all bulk links of all the corresponding link observables.

4.2. Statistics

Since the boundary conditions vary with time in our simulations, the system does not settle into a unique stationary state. In particular, the ergodic theorem does not apply, so that time averages do not converge to stationary expectations. We therefore repeated each simulation n times, and for each value of t we estimated $\langle X_t \rangle$ via

$$\frac{1}{n}\sum_{i=1}^{n} X_t^{(i)}$$

where $X_t^{(i)}$ is the realization of X_t obtained during the ith run. Here X_t might be the density of a link, or the space-mean velocity of a link, or indeed any of the observables mentioned in the previous section. In this way, for a given observable, X_t, we estimate the *average process* $\langle X_1 \rangle, \langle X_2 \rangle \ldots$. All results in this paper are based on $n = 100$ simulation runs.

4.3. Travel times

In a given simulation, for each value of t we have a list $\mathcal{T}_t^{(1)}, \mathcal{T}_t^{(2)}, \ldots, \mathcal{T}_t^{(k_t)}$ where k_t is the number (possibly zero) of vehicles to leave the network at time t, and $\mathcal{T}_t^{(i)}$ is the total amount of time spent in the network by the ith such vehicle. In a simulation of duration T iterations, the total number of vehicles that have traversed the network is therefore

$$\sum_{t=1}^{T} k_t.$$

For a given simulation, we compute the mean total travel time per vehicle

$$m_{\mathcal{T}} = \frac{\sum_{t=1}^{T}\sum_{i=1}^{k_t} \mathcal{T}_t^{(i)}}{\sum_{t=1}^{T} k_t}$$

and its fluctuation

$$s_{\mathcal{T}}^2 = \frac{\sum_{t=1}^{T}\sum_{i=1}^{k_t} (\mathcal{T}_t^{(i)} - m_{\mathcal{T}})^2}{\sum_{t=1}^{T} k_t}.$$

We emphasize that $m_{\mathcal{T}}$ and $s_{\mathcal{T}}$ are random variables. We again estimate the averages, $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$, by simply measuring $m_{\mathcal{T}}$ and $s_{\mathcal{T}}$ in n independent simulations and computing their arithmetic means. It seems intuitively reasonable that both $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ provide useful measures of network efficiency.

5. Simulations – Kew

5.1. Empirical data

A section of the Melbourne suburb of Kew consisting of fourteen signalized intersections was chosen as the network on which to test our cellular automaton; see figure 1. This network corresponds to the directed graph shown in figure 2. A list of nodes and links is input into the model in order to define the actual network. For each link the length, number of lanes, and speed limit must also be input, and for each node a list of phases must be provided. The phases input into the model are simplified versions of the actual phases used by SCATS, which ignore complications such as trams and pedestrians that are currently not taken into account in our model.

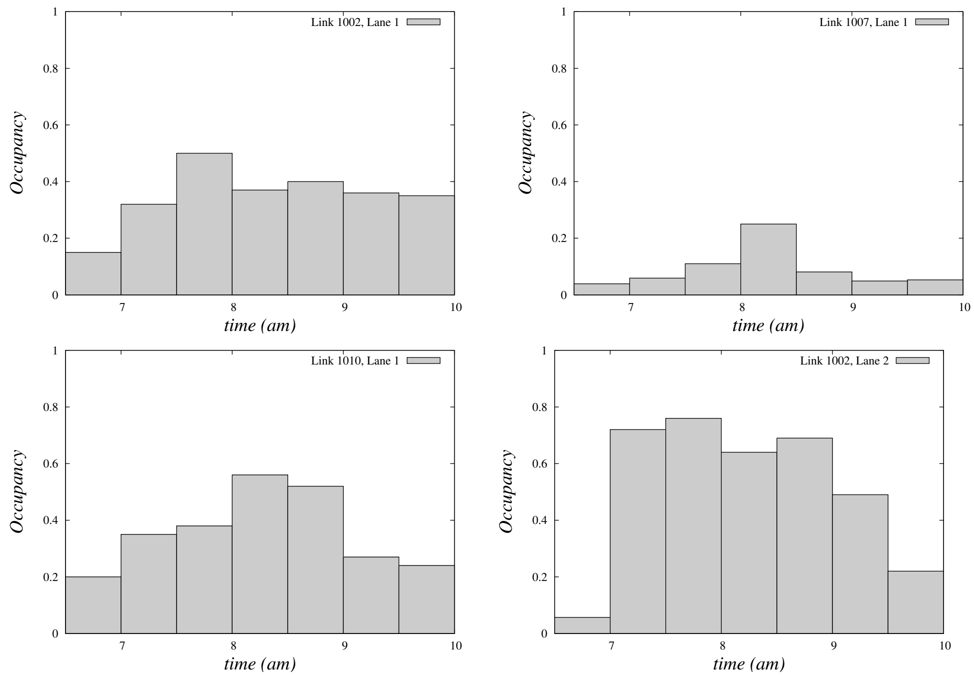


Figure 5. Smoothed time series of the empirical SCATS occupancies, into 30 minute bins, on some representative boundary in-lanes of the Kew network. Link labels correspond to the labels in figure 2.

5.1.1. Boundary Conditions In addition, suitable boundary conditions need to be applied to the model, and an initial configuration needs to be specified. We use boundary data in our model for two distinct purposes; as a means to correctly control the inflow and outflow of vehicles from the network, and also as a means of informing adaptive signal decisions at intersections on the boundary of the network.

We use time inhomogeneous boundary conditions as explained in section 2.4, and empirical SCATS stop-line occupancies to implement the inflow rates as described in section 2.3. For each boundary in-lane of the network in figure 2, VicRoads provided us with a time series of the stop-line occupancy, with the exception of link 1012. In this case a three-lane link was covered by a single detector, making it impossible to obtain reliable data. For this link, we have used heuristically reasonable inflow rates based on simulations by the origin-destination software package MITM (see section 5.1.2 for more on MITM).

The occupancy time series provided by VicRoads was for the period 6:30am to 10:00am, in time intervals of 1 minute. To remove the effect of fluctuations due to traffic cycles, typically taking between two and three minutes, we smoothed this data into bins of 30 minutes, implying that $T_B = 1800$ iterations (simulated seconds) in our simulations (see section 2.4 for the definition of T_B). Smaller bins resulted in very noisy profiles. Figure 5 shows some examples of the data used. As described in section 2.3, at each instant of time, we use the stop-line occupancy, o_λ to set the input probability, α_λ, into boundary in-lane λ.

In addition, for each boundary in-lane and each boundary out-lane, we require estimates of the total density, ρ_λ, in order to set the SOTL demands according to the

demand function (5). In the absence of any detailed empirical data for this quantity, we simply made the assumption that $\rho_\lambda \approx o_\lambda$, for boundary in-lanes, and arbitrarily set $\rho_\lambda = 0$ for the case of boundary out-lanes. This is almost surely an overestimate of ρ_λ in the case of in-lanes.

In all our simulations we started from an empty network, letting the system fill up using the time inhomogeneous boundary conditions.

5.1.2. Turning probabilities For each node in the chosen network, VicRoads provided simulated data from the MITM software package that lists predicted volumes through each $(inlink, outlink)$ pair, over a period of one hour. In order to estimate the required turning probabilities described in section 2 we computed turning ratios from these simulated volumes. We note that MITM is designed for city-wide demographic simulations, and so by using it to obtain turning probabilities at specific intersections, we are likely using it to answer questions on a spatial resolution beyond its designed accuracy. While the resulting values of the turning probabilities may therefore differ from reality, we do expect them to be at least indicative of the true results, for most intersections. In order to obtain more accurate turning probabilities, the MITM data could in principle be compared/augmented with SCATS data where available, however SCATS occupancies are not sufficient to obtain all the required turning ratios.

5.2. Simulations

5.2.1. Comparing SOTL vs fixed-cycle traffic lights. For the observables ρ_l, Q_l, J_l and v_l defined in section 4, the left column of figure 6 shows typical examples of the average processes on an **uncongested** link (18 in figure 2), using SOTL with demand function (5) and demand exponents $(m, n) = (1, 1)$, threshold $\theta = 2$. It seems that during each inflow epoch a new stationary state is reached, before the link is perturbed out of that state and into another one when the inflow probabilities are changed. This behaviour is clearly visible in the density plot, but is also apparent in the queue length and flow plots, and to a lesser extent in the speed plot.

For comparison, we repeated the simulations using non-adaptive fixed-cycle traffic lights (for a definition see Appendix A.6) and measured the same observables. These are plotted in the right column of figure 6. The green times for each phase in this case were obtained from the actual SCATS values during morning peak hour (7am–9am). Clearly, the fluctuations in these results are much larger than for SOTL. A similar observation was made by Lämmer and Helbing in [27], who studied self-organizing traffic lights using a fluid-dynamic model for the traffic flow in urban road networks.

Figure 7 shows typical examples of the average processes on a **congested** link (7 in figure 2), both for SOTL with demand function (5) and demand exponents $(m, n) = (1, 1)$, threshold $\theta = 2$, as well as for fixed-cycle traffic lights. Unlike the uncongested plots it does not appear that stationarity is reached within the individual inflow epochs, suggesting that relaxation towards stationarity is much faster at low density than at high density. However, the transitions between inflow epochs are still visible in the density plot for SOTL.

Figure 8 shows typical examples of the average network-mean processes, ρ, Q, J and v, both for SOTL and fixed-cycle traffic lights. There are jump-discontinuities in the SOTL evolutions of Q and J at times up to about $t = 7000$, suggesting that these system-wide observables manage to reach their stationary value within each of these early epochs. These jumps are less pronounced in ρ and v. For later times, the jumps

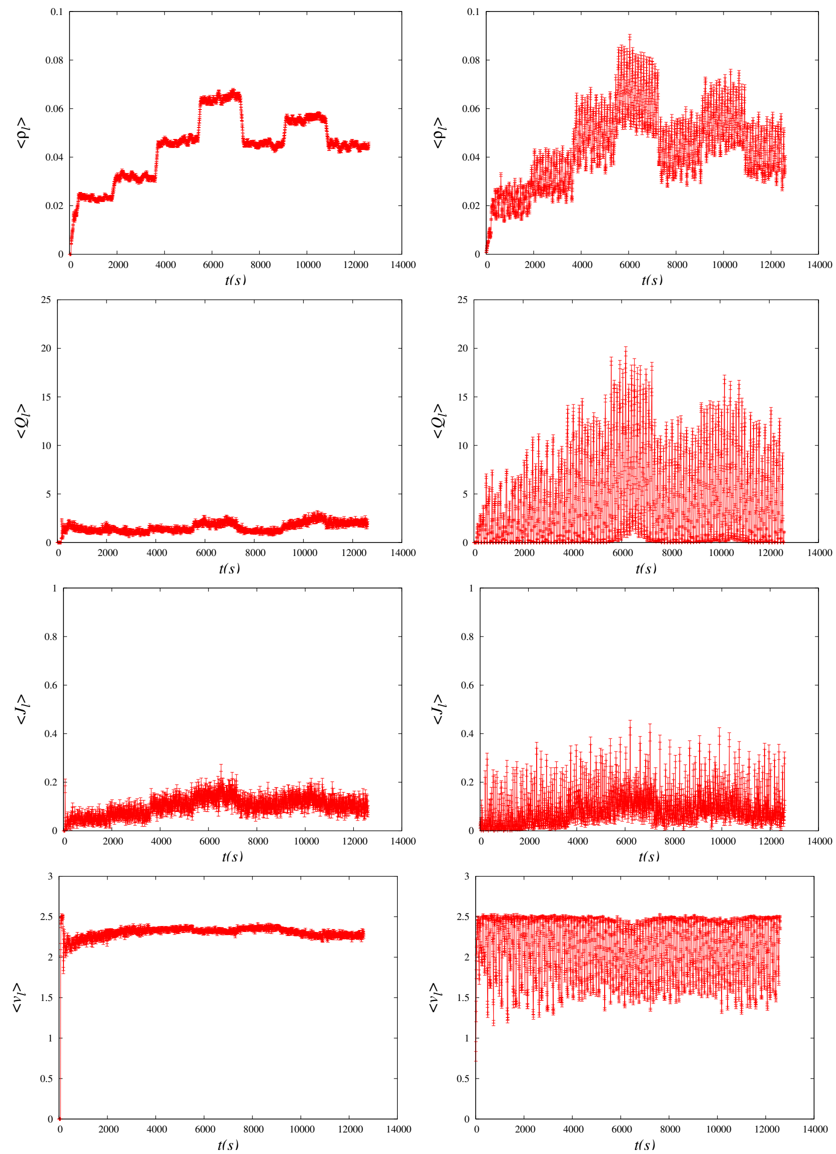


Figure 6. Uncongested evolution. From top: SOTL (left) vs Fixed Cycle (right) evolution of the density, queue length, flow and space-mean speed, on an uncongested link in the Kew network (link 18 in figure 2). The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$. Time inhomogeneous boundary conditions based on SCATS data were imposed, such as in figure 5.

in any of the network observables are much less significant, suggesting that during later epochs, at the network level the system never really reaches stationarity. As the network is relatively uncongested during early epochs, this confirms that relaxation towards stationarity is much faster at low density than at high density.

In summary, we find that SOTL gives better results than fixed-cycle traffic lights

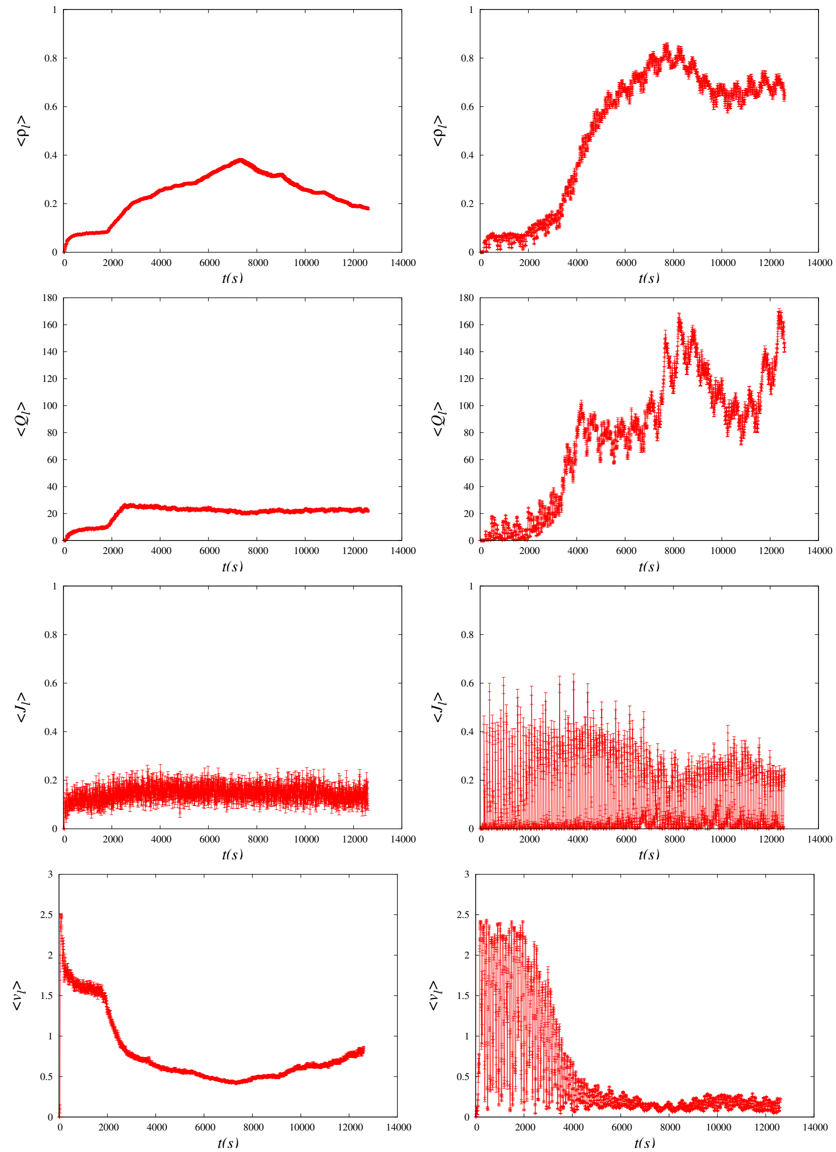


Figure 7. Congested evolution. From top: SOTL (left) vs Fixed Cycle (right) evolution of the density, queue length, flow and space-mean speed, on a congested link in the Kew network (link 7 in figure 2). The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$. Time inhomogeneous boundary conditions based on SCATS data were imposed, such as those shown in figure 5.

for the means of the density, queue length, flow and speed. Furthermore, in all cases SOTL produces much smaller fluctuations. Moreover, at later times, when the network is congested but the boundary inflow decreases, SOTL allows the system to adjust more rapidly to the changed boundary conditions.

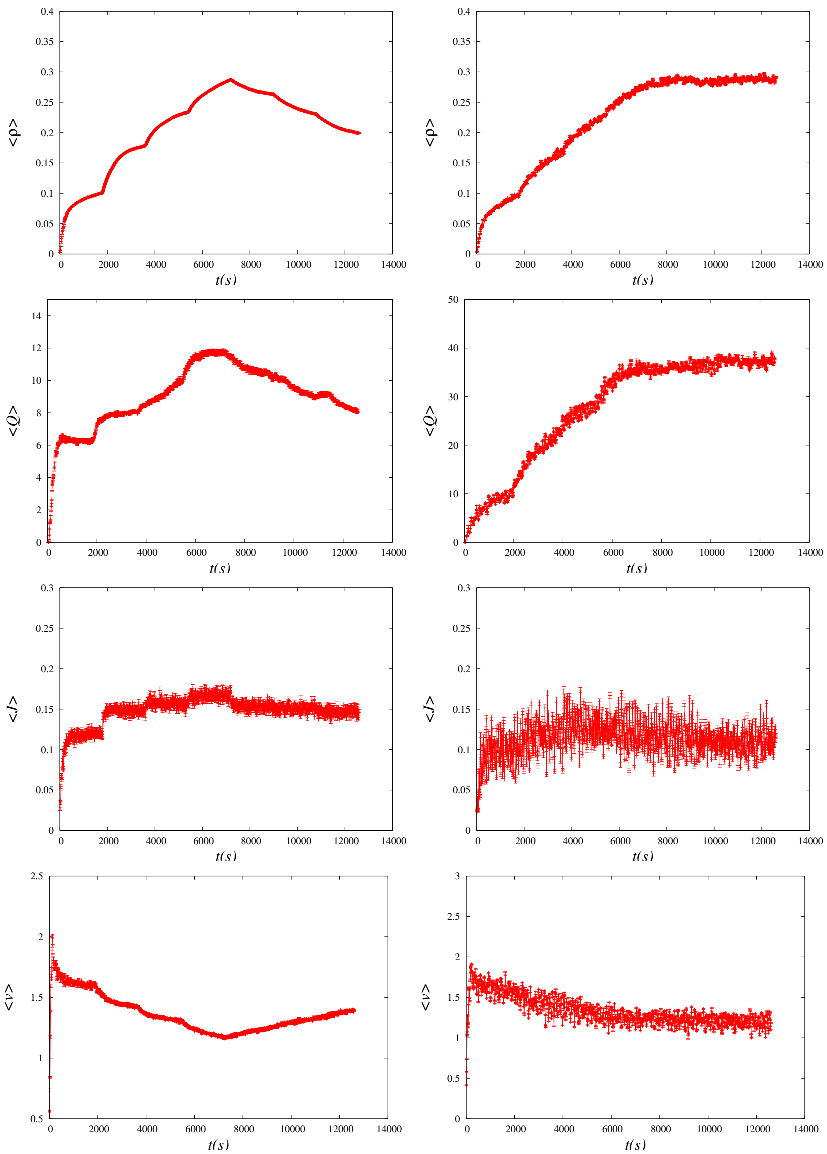


Figure 8. Network means. From top: SOTL (left) vs Fixed Cycle (right) evolution of the network-averaged density, space-mean speed, flow and queue length in the Kew network. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$. Time inhomogeneous boundary conditions based on SCATS data were imposed, such as those shown in figure 5.

5.2.2. Comparing upstream-only vs upstream-downstream SOTL. The average values of the travel time $m_{\mathcal{T}}$ and its fluctuation $s_{\mathcal{T}}$ are presented in figure 9 as a function of the SOTL threshold parameter θ for the two choices of exponents $(m, n) = (1, 0)$ and $(m, n) = (1, 1)$ in the SOTL demand function (5). The former choice corresponds to an upstream-only version of SOTL, while the latter corresponds to a hybrid upstream-

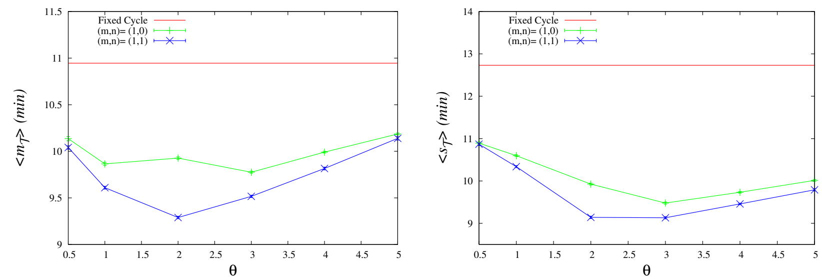


Figure 9. Mean travel time $\langle m_{\mathcal{T}} \rangle$ and its fluctuation $\langle s_{\mathcal{T}} \rangle$ vs SOTL threshold parameter θ, for the Kew network, with the SOTL demand function (5) and SOTL demand exponents $(m, n) = (1, 0), (1, 1)$. The horizontal line shows the corresponding value for the system with fixed-cycle traffic lights.

downstream version. For comparison, we include in the figures the corresponding values of $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ for the network with fixed-cycle traffic lights, which are independent of θ. We begin by noting that both SOTL strategies result in significantly lower values of both $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ than fixed-cycle traffic lights. The adaptive systems therefore not only outperform the fixed-cycle system on average, but they are also more reliable. The observation that SOTL produces smaller fluctuations for the vehicle travel time is entirely consistent with the behaviour presented in figures 6, 7 and 8. The $(1, 1)$ model appears to have an optimal value of θ near $\theta \approx 2$, in terms of both $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$. In both cases, there is range of θ around $1 \leq \theta \leq 3$ for which the dependence of $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ on θ appears weak. This suggests that the SOTL methodology is reasonably robust with respect to the parameter θ.

Note also that for every value of θ the values of both $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ for the $(m, n) = (1, 1)$ model are lower than the corresponding values for the $(1, 0)$ model. While the difference is not large, it is clearly statistically significant; see Table 1. Therefore, we can cautiously conclude that the $(1, 1)$ model is marginally more efficient (smaller travel times) and more reliable (smaller fluctuations in travel times) than the $(1, 0)$ model, for this network with the given boundary conditions. To produce a loose

Table 1. Numerical values of the mean $\langle m_{\mathcal{T}} \rangle$ and fluctuation $\langle s_{\mathcal{T}} \rangle$ of the vehicle travel time for the simulations of the $(1, 0)$ and $(1, 1)$ models on the Kew network. The statistical error shown corresponds to one standard deviation. The units are minutes. For comparison, the corresponding values using fixed-cycle traffic lights are $\langle m_{\mathcal{T}} \rangle_{\text{fc}} = 10.95 \pm 0.02$ and $\langle s_{\mathcal{T}} \rangle_{\text{fc}} = 12.73 \pm 0.07$.

	$(m, n) = (1, 0)$		$(m, n) = (1, 1)$	
θ	$\langle m_{\mathcal{T}} \rangle$	$\langle s_{\mathcal{T}} \rangle$	$\langle m_{\mathcal{T}} \rangle$	$\langle s_{\mathcal{T}} \rangle$
0.5	10.14 ± 0.02	10.89 ± 0.03	10.04 ± 0.02	10.87 ± 0.03
1.0	9.87 ± 0.02	10.60 ± 0.02	9.61 ± 0.01	10.34 ± 0.02
2.0	9.93 ± 0.01	9.93 ± 0.02	9.29 ± 0.01	9.14 ± 0.03
3.0	9.78 ± 0.01	9.48 ± 0.03	9.52 ± 0.01	9.13 ± 0.03
4.0	9.99 ± 0.01	9.73 ± 0.02	9.82 ± 0.01	9.46 ± 0.02
5.0	10.19 ± 0.01	10.01 ± 0.02	10.14 ± 0.02	9.79 ± 0.02

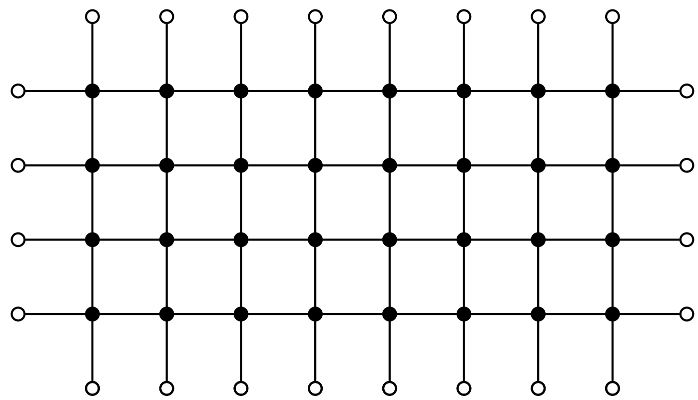


Figure 10. Square grid with $L_x = 8$ and $L_y = 4$. The nodes in the network are represented by the full circles, while empty circles represent boundary nodes. Each link in the figure actually corresponds to two directed edges (one in each direction), each consisting of two lanes.

estimate of the relative performance of the two models we note that

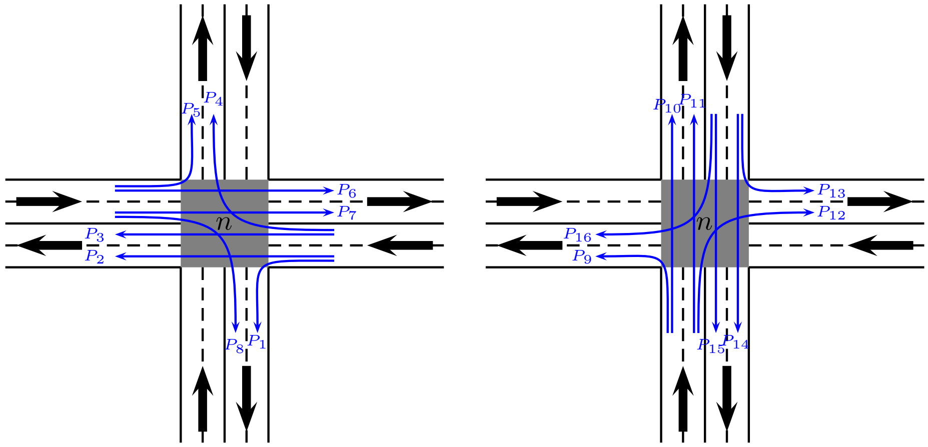
$$\frac{\min \langle m_{\mathcal{T}} \rangle_{(1,0)} - \min \langle m_{\mathcal{T}} \rangle_{(1,1)}}{\min \langle m_{\mathcal{T}} \rangle_{(1,0)}} \approx 5\%,$$
$$\frac{\min \langle s_{\mathcal{T}} \rangle_{(1,0)} - \min \langle s_{\mathcal{T}} \rangle_{(1,1)}}{\min \langle s_{\mathcal{T}} \rangle_{(1,0)}} \approx 4\%.$$

6. Simulations – Square grid

In addition to testing SOTL on the Kew network, we also tested it on a regular $L_x \times L_y$ square grid, as shown in figure 10. Each link in the network was given two lanes, and for simplicity we did not include turning lanes. Each node was given four phases; an east/west phase, a north/south phase, and two corresponding turning phases (corresponding to green turn-arrows and red lights), see figure 11. When we used fixed-cycle traffic lights, the ordering was east/west, turning, north/south, turning, east/west.. . . etc., corresponding to cyclically repeating the phases $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ and $\mathcal{P}_4$. The lengths of all bulk links were set to 300 metres, which is of the order of a city block in Melbourne's CBD. We simulated the case $L_x = L_y = 4$, which is approximately the same size as the Kew network. It is of significant interest, however, to study the effect of varying L_x and L_y. We intend to pursue this in future studies.

Let us define T to be the total duration of the simulation. To ensure the square-grid simulations were analogous to the Kew simulations we again chose to simulate for $3\frac{1}{2}$ hours, so that $T = 12,600s$. We also emulated the effect of the AM peak hour by choosing time dependent boundary conditions, as shown in figure 12.

More precisely, on each boundary lane λ we imposed the following time-dependent



(a) $\mathcal{P}_1 = \{P_1, P_2, \ldots P_8\}$ (b) $\mathcal{P}_3 = \{P_9, P_{10}, \ldots P_{16}\}$

Figure 11. The phases used in the square-grid network were the east/west phase $\mathcal{P}_1 = \{P_1, P_2, \ldots, P_8\}$, the north/south phase $\mathcal{P}_3 = \{P_9, P_{10}, \ldots, P_{16}\}$, and the two corresponding turning phases $\mathcal{P}_2 = \{P_1, P_4, P_5, P_8\}$ and $\mathcal{P}_4 = \{P_9, P_{12}, P_{13}, P_{16}\}$.

density profile:

$$\rho_\lambda(t) = \begin{cases} \left(\dfrac{\rho_{\lambda,\max} - \rho_{\lambda,\min}}{T_g}\right) t + \rho_{\lambda,\min} & 0 \le t < T_g \\ \rho_{\lambda,\max} & T_g \le t \le T - T_g \\ \left(\dfrac{\rho_{\lambda,\max} - \rho_{\lambda,\min}}{T_g}\right)(T - t) + \rho_{\lambda,\min} & T - T_g < t \le T. \end{cases} \quad (6)$$

The parameter T_g is the amount of time that the density profile spent *growing*, before plateauing at its maximum value, $\rho_{\lambda,\max}$. In all our simulations we set $T_g = 3600s$, i.e. 1 hour. Note that because we have assumed the profile is symmetric, T_g is also the amount of time that the profile spends decaying after its plateau.

There are two remaining free parameters in (6), $\rho_{\lambda,\max}$ and $\rho_{\lambda,\min}$. We ran simulations of the above networks under three different scenarios of $(\rho_{\lambda,\max}, \rho_{\lambda,\min})$, corresponding to uniform low density, uniform high density, and a strong westbound bias. The precise values used in each of these scenarios are described in the sections to follow. We note that, analogously to the simulations we performed on the Kew network, we chose to bin the profile (6) into bins of $T_B = 30$ minute duration. While binning is necessary for smoothing empirical data, as used in the Kew simulations discussed in Section 5, it is in principle not necessary here; our motivation for using binning here was simply to avoid introducing irrelevant differences between the Kew and square-grid networks. In section 6.2 we discuss in detail the effects of choosing different values of the binning time T_B.

At each instant of time, the value of $\rho_\lambda(t)$ was used to inform the SOTL demand function (5) for nodes adjacent to boundary links. Furthermore, we chose to set the

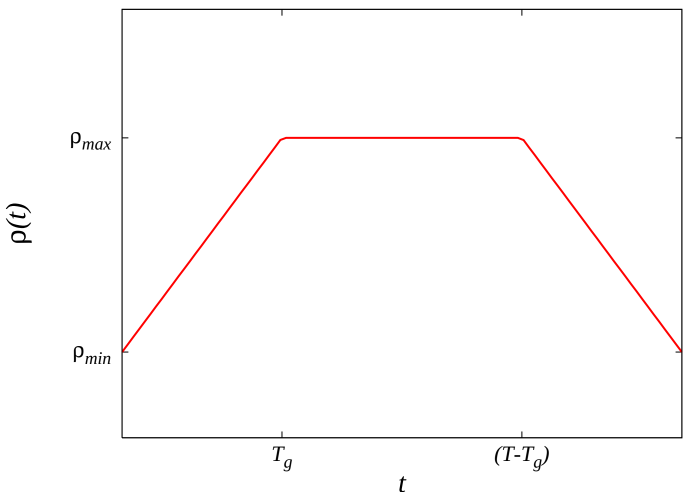


Figure 12. Time-dependent density profile used in the square-grid simulations. Cf. equation (6).

lengths of boundary in-lanes to 150 metres (half the value of the bulk lanes), which allowed us to legitimately set the boundary input probability to be $\alpha_\lambda = \rho_\lambda$. See section 2.3. For boundary out-lanes we again simply set $\rho_{\lambda,1} = 0$.

Finally, for each intersection we had to set the following twelve turning probabilities,

$$\begin{pmatrix} p_{WW} & p_{WN} & p_{WS} \\ p_{EE} & p_{EN} & p_{ES} \\ p_{NN} & p_{NW} & p_{NE} \\ p_{SS} & p_{SW} & p_{SE} \end{pmatrix},$$

where p_{NW} is the probability that a northbound vehicle chooses to turn onto a westbound link at the approaching intersection, and the other eleven parameters are defined analogously in the obvious way. We chose turning probabilities in a way that was consistent with each of the above three boundary profile scenarios. The precise values in each case are described below.

6.1. Westbound bias in the boundary conditions

To produce strong westward bias we set $\rho_{\lambda,\min} = 0.1$ for all boundary in-lanes λ, and set $\rho_{\lambda,\max} = 0.4$ for westbound in-lanes and $\rho_{\lambda,\max} = 0.2$ for all the other in-lanes.

The turning probabilities were also chosen to impose a westward bias, as follows:

$$\begin{pmatrix} p_{WW} & p_{WN} & p_{WS} \\ p_{EE} & p_{EN} & p_{ES} \\ p_{NN} & p_{NW} & p_{NE} \\ p_{SS} & p_{SW} & p_{SE} \end{pmatrix} = \begin{pmatrix} 0.6 & 0.2 & 0.2 \\ 0.34 & 0.33 & 0.33 \\ 0.34 & 0.33 & 0.33 \\ 0.34 & 0.33 & 0.33 \end{pmatrix}. \qquad (7)$$

6.1.1. Comparing SOTL vs fixed-cycle traffic lights. In figures 13 and 14 we show plots of the link observables ρ_l, v_l, Q_l and J_l, on westbound and northbound bulk links. Due to the symmetry of the boundary conditions southbound links behave identically

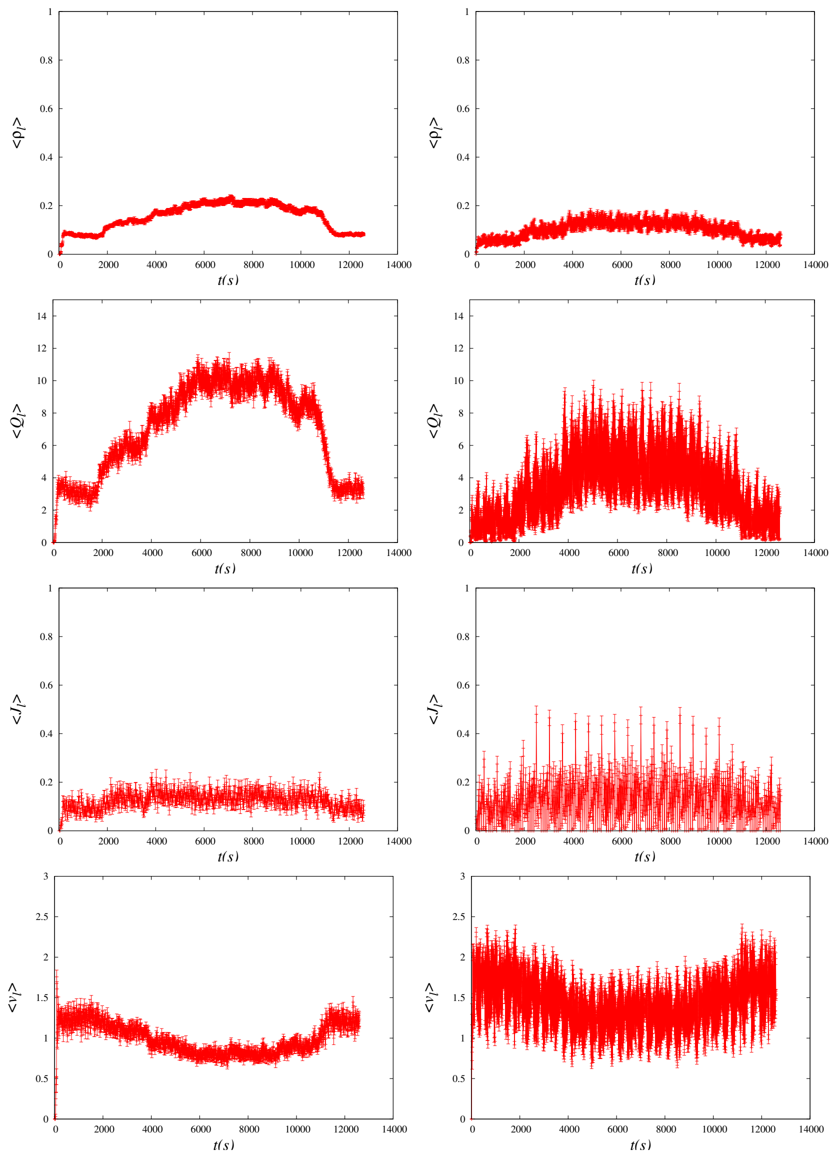


Figure 13. Westbound Bias. From top: SOTL (left) vs Fixed Cycle (right) evolution of the density, queue length, flow and space-mean speed, on a given westbound link for the westbound-biased 4×4 square grid. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$. Time inhomogeneous boundary conditions of figure 12 were imposed.

to northbound links, and we find that also eastbound links behave similarly. We compare SOTL (left column) with $(m, n) = (1, 1)$ and $\theta = 2$, vs fixed-cycle traffic lights (right column). The fixed green time of each phase used in the fixed-cycle simulation was determined from the corresponding SOTL values midway through the morning peak hour. We further note that since we are binning the boundary inflows

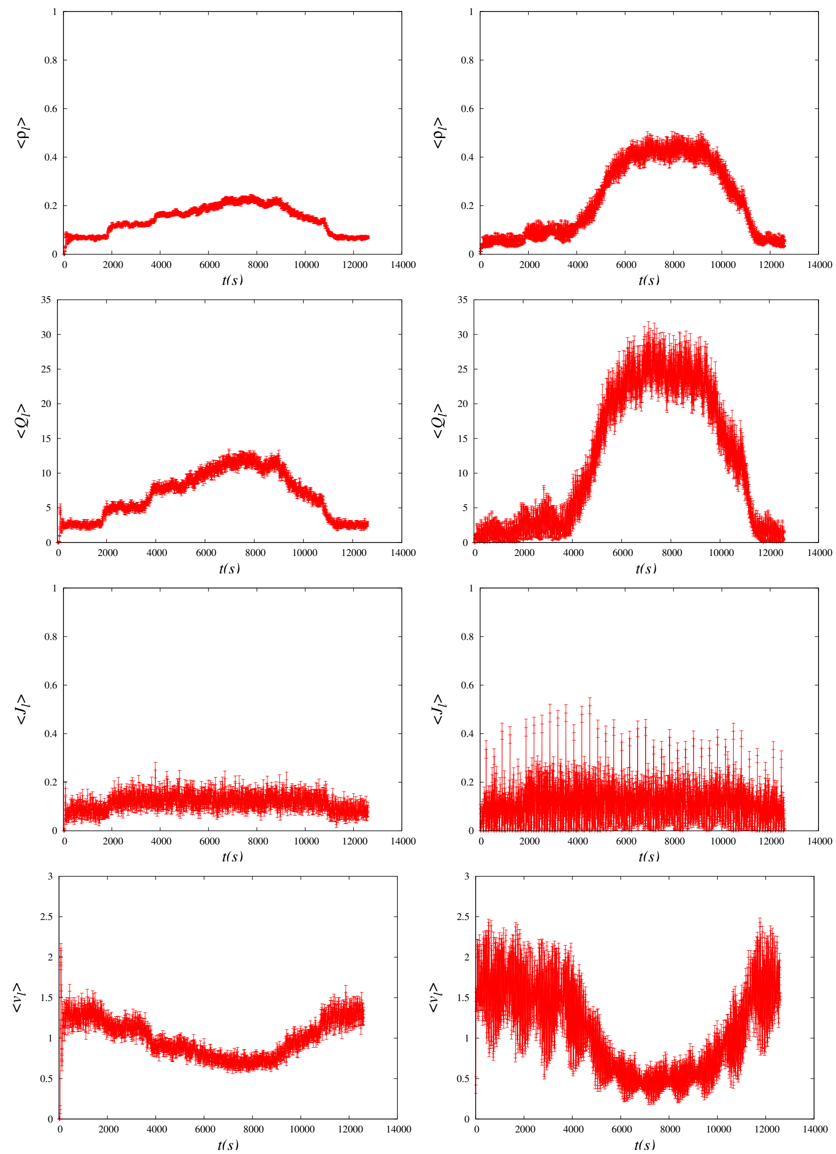


Figure 14. Westbound Bias. From top: SOTL (left) vs Fixed Cycle (right) evolution of the density, queue length, flow and space-mean speed, on a given northbound link for the westbound-biased 4×4 square grid. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$. Time inhomogeneous boundary conditions of figure 12 were imposed.

in the same way as for Kew, i.e. the boundary inflows change every 1800 second, the profiles show artificial jumps as a result.

For the westbound link, the means for fixed-cycle traffic lights are comparable to SOTL, and in some cases even marginally better. However, they are considerably worse for the northbound link. In both cases, the fluctuations for SOTL are much

smaller than those for fixed-cycle traffic lights. Furthermore, the northbound link, being less congested, adjusts more rapidly to the changing boundary conditions at later times than the westbound link.

6.1.2. Comparing upstream-only vs upstream-downstream SOTL. The average values of the travel time $m_{\mathcal{T}}$ and its fluctuation $s_{\mathcal{T}}$ are presented in figure 15. Both the $\langle m_{\mathcal{T}} \rangle$

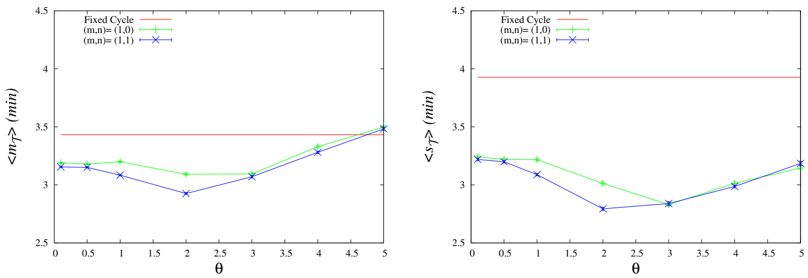


Figure 15. Mean travel time $\langle m_{\mathcal{T}} \rangle$ and its fluctuation $\langle s_{\mathcal{T}} \rangle$ vs SOTL threshold parameter θ, for the westbound-biased 4×4 square grid, with the SOTL demand function (5) and SOTL demand exponents $(m, n) = (1, 0), (1, 1)$. The horizontal line shows the corresponding value for the system with fixed-cycle traffic lights.

and $\langle s_{\mathcal{T}} \rangle$ curves appear to have an optimal value around $\theta \approx 2$ for the $(m, n) = (1, 1)$ system, and a slightly larger optimal value around $\theta \approx 3$ for the $(1, 0)$ system. For both SOTL systems, the curves for both $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ lie well below the horizontal line corresponding to fixed-cycle traffic lights, except for large θ. Furthermore, just as we found for the Kew network, for every value of θ the values of $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ for the $(m, n) = (1, 1)$ model provide lower bounds on the corresponding value for $(1, 0)$, to within statistical errors. For small and large θ the difference between the $(1, 0)$ and $(1, 1)$ models does not appear to be statistically significant, but it certainly does appear to be statistically significant for $1 \le \theta \le 2$. See Table 2. Therefore, we can conclude that the $(1, 1)$ model is again both more efficient and more reliable than the $(1, 0)$ model. To quantify this approximately, we note that

$$\frac{\min \langle m_{\mathcal{T}} \rangle_{(1,0)} - \min \langle m_{\mathcal{T}} \rangle_{(1,1)}}{\min \langle m_{\mathcal{T}} \rangle_{(1,0)}} \approx 5\%,$$

$$\frac{\min \langle s_{\mathcal{T}} \rangle_{(1,0)} - \min \langle s_{\mathcal{T}} \rangle_{(1,1)}}{\min \langle s_{\mathcal{T}} \rangle_{(1,0)}} \approx 1\%.$$

6.2. Effect of varying the binning time

As noted previously, we chose to bin the profile (6) into bins of $T_B = 30$ minutes, in order to avoid introducing irrelevant differences between the Kew and square-grid networks. From the perspective of the square lattice, however, the choice $T_B = 30$ is essentially arbitrary, so in this section we investigate the effect of varying the value of T_B. This is of interest for the following reason. As can be seen from the plateaus in figures 6 and 8, as well as in figures 13 and 14, when using $T_B = 30$ minutes, many observables relax to approximate stationarity within each individual inflow epoch. In real traffic situations however, the input rates may change on a faster time scale. In

addition, for larger networks the relaxation time would be larger, and the network may only reach stationarity on a time scale larger than $T_B = 30$ minutes. We therefore studied the effect of using smaller values of T_B, within which the network is not able to reach stationarity.

In figure 16 we plot the link observables for a northbound link on the westbound-biased square grid with $T_B = 5$ minutes. It can be seen that in contrast to the corresponding figure 14 for $T_B = 30$ minutes, the profiles do not plateau, and hence this link does not reach stationarity within the inflow epoch of 5 minutes. Similar behaviour was observed on the other links in the network. It seems natural, therefore, to expect that the dependence of $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ on θ displayed in figure 15 may be modified when $T_B = 5$ minutes. Figure 17 shows that this is not the case; the plots of $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ for the system with $T_B = 5$ minutes shown in figure 17 are in fact qualitatively the same as those in figure 15 for the system with $T_B = 30$ minutes. [+] Similar results were also observed for the $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ plots when using $T_B = 10$ and $T_B = 15$ minutes. In summary, we have found strong evidence that the shape of the $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ plots are quite robust to changes in the parameter T_B, regardless of whether T_B is small or large compared to the relaxation time of the network.

6.3. Uniform high-density boundary conditions

To produce uniform high-density boundary conditions, for each boundary in-lane we set $\rho_{\lambda,\max} = 0.8$ and $\rho_{\lambda,\min} = 0.2$, and the turning probabilities were chosen to be

$$\begin{pmatrix} p_{WW} & p_{WN} & p_{WS} \\ p_{EE} & p_{EN} & p_{ES} \\ p_{NN} & p_{NW} & p_{NE} \\ p_{SS} & p_{SW} & p_{SE} \end{pmatrix} = \begin{pmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0.25 & 0.25 \\ 0.5 & 0.25 & 0.25 \\ 0.5 & 0.25 & 0.25 \end{pmatrix}. \tag{8}$$

These turning probabilities imply that regardless of a vehicle's direction, it chooses to continue straight with probability 1/2, and turn either left or right with probability 1/4.

[+] We note that using our particular binning procedure, the area under the input profile in figure 6 is slightly larger when using a discretization of $T_B = 5$ minute bins than when using $T_B = 30$ minute bins. The slight upward shift in the travel time profiles for the $T_B = 5$ simulations relative to the $T_B = 30$ simulations can therefore be understood as a simple consequence of a slightly higher total volume of vehicles that enter the network during the simulation.

Table 2. Numerical values of the mean $\langle m_{\mathcal{T}} \rangle$ and fluctuation $\langle s_{\mathcal{T}} \rangle$ of the vehicle travel time for the westbound-biased simulations of the $(1,0)$ and $(1,1)$ models. The statistical error shown corresponds to one standard deviation. The units are minutes. For comparison, the corresponding values using fixed-cycle traffic lights are $\langle m_{\mathcal{T}} \rangle_{\rm fc} = 3.43 \pm 0.01$ and $\langle s_{\mathcal{T}} \rangle_{\rm fc} = 3.93 \pm 0.02$.

	$(m,n) = (1,0)$		$(m,n) = (1,1)$	
θ	$\langle m_{\mathcal{T}} \rangle$	$\langle s_{\mathcal{T}} \rangle$	$\langle m_{\mathcal{T}} \rangle$	$\langle s_{\mathcal{T}} \rangle$
0.1	3.19 ± 0.01	3.24 ± 0.01	3.15 ± 0.01	3.22 ± 0.01
0.5	3.18 ± 0.01	3.22 ± 0.01	3.15 ± 0.01	3.20 ± 0.01
1.0	3.20 ± 0.01	3.22 ± 0.01	3.08 ± 0.01	3.09 ± 0.01
2.0	3.09 ± 0.01	3.01 ± 0.01	2.93 ± 0.01	2.79 ± 0.01
3.0	3.09 ± 0.01	2.83 ± 0.01	3.07 ± 0.01	2.84 ± 0.02
4.0	3.33 ± 0.01	3.01 ± 0.02	3.28 ± 0.01	2.99 ± 0.01
5.0	3.50 ± 0.01	3.15 ± 0.01	3.48 ± 0.01	3.18 ± 0.02

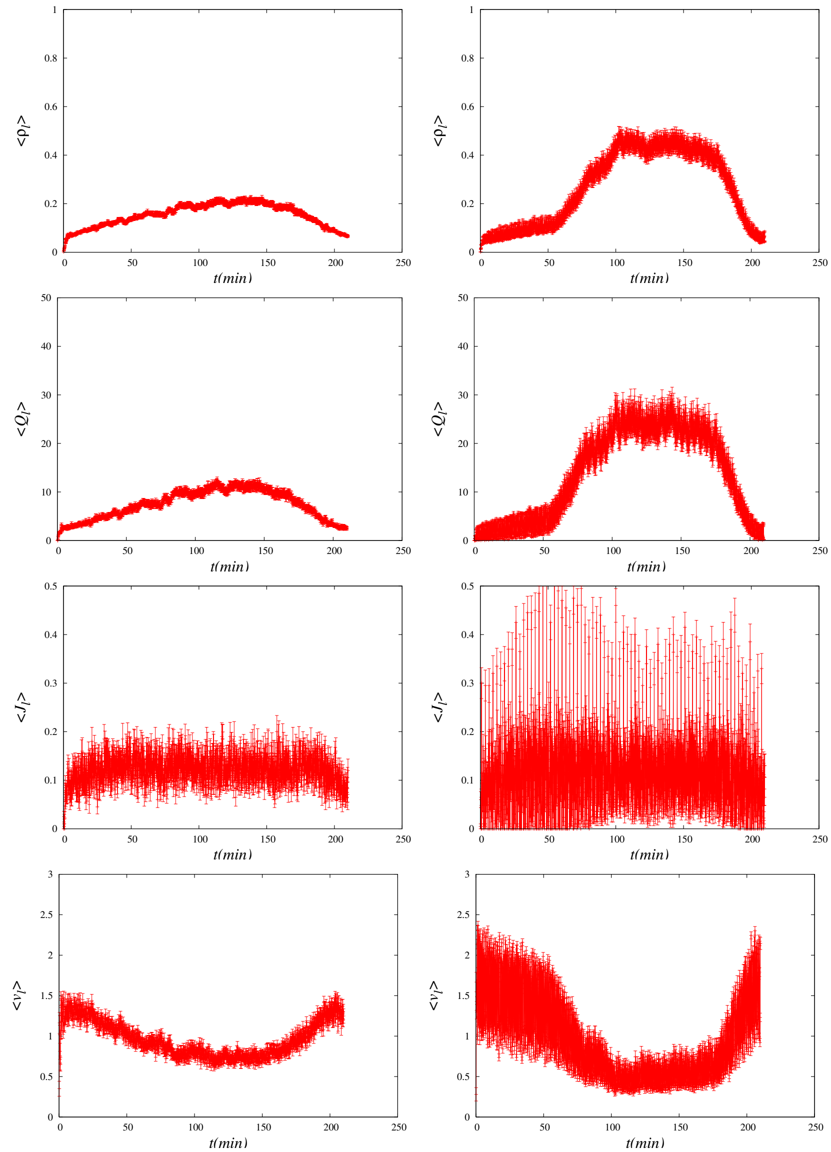


Figure 16. Westbound Bias. From top: SOTL (left) vs Fixed Cycle (right) evolution of the density, queue length, flow and space-mean speed, on a given northbound link for the westbound-biased 4×4 square grid with $T_B = 5$. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$. Time inhomogeneous boundary conditions of figure 12 were imposed.

6.3.1. Comparing SOTL vs fixed-cycle traffic lights. In figure 18 we compare the evolution of the link observables ρ_l, Q_l, J_l and v_l for the high-density square-lattice network for the adaptive SOTL update vs fixed-cycle traffic lights. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$. Again, the values of the fixed green times in the fixed cycle simulations

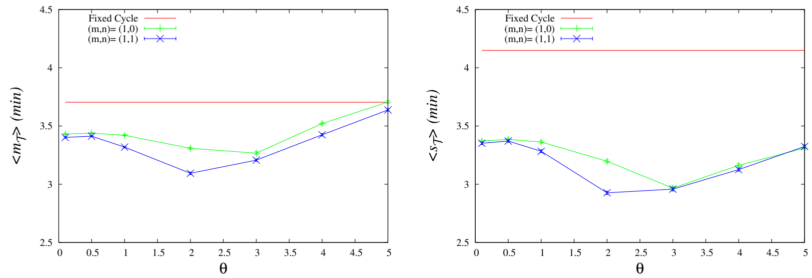


Figure 17. Mean travel time $\langle m_{\mathcal{T}} \rangle$ and its fluctuation $\langle s_{\mathcal{T}} \rangle$ vs SOTL threshold parameter θ, for the westbound-biased 4×4 square grid using $T_B = 5$, with the SOTL demand function (5) and SOTL demand exponents $(m, n) = (1, 0), (1, 1)$. The horizontal line shows the corresponding value for the system with fixed-cycle traffic lights.

were determined from the corresponding SOTL values midway through the morning peak hour. As for the Kew network studied in section 5, SOTL clearly performs better, in particular the density and queue lengths of SOTL are significantly lower than for fixed-cycle traffic lights, while the flow is larger. Moreover, at later times when the network is congested but the boundary inflow decreases, SOTL allows the system to adjust more rapidly to the changed boundary conditions. Finally, the fluctuations produced by SOTL are again much smaller than those for fixed-cycle traffic lights. In figure 19 we compare the evolution of their network averages, which show similar behaviour.

6.3.2. Comparing upstream-only vs upstream-downstream SOTL. The average values of the travel time $m_{\mathcal{T}}$ and its fluctuation $s_{\mathcal{T}}$ are presented in figure 20. Unlike the behaviour displayed in figures 9 and 15, there does not appear to be an optimal value of θ for the $\langle m_{\mathcal{T}} \rangle$ curve for the $(m, n) = (1, 1)$ model, and in fact the curve is only rather weakly dependent on θ. Another interesting feature is that although the fixed-cycle traffic lights (whose green times were chosen by analysing simulated green times from SOTL simulations using the $(m, n) = (1, 1)$ model) are less efficient than $(1, 1)$-SOTL for all the θ we studied, they are *more* efficient than $(1, 0)$-SOTL for all $\theta \geq 3$.

Once again, for every value of θ the values of $\langle m_{\mathcal{T}} \rangle$ and $\langle s_{\mathcal{T}} \rangle$ for the $(m, n) = (1, 1)$ model are lower than the corresponding value for the $(1, 0)$ model. The difference is statistically significant, except possibly for $\theta < 1$; see Table 3. Therefore, we again conclude that the $(1, 1)$ model is marginally more efficient and more reliable than the $(1, 0)$ model. To approximately quantify this we note that

$$\frac{\min \langle m_{\mathcal{T}} \rangle_{(1,0)} - \min \langle m_{\mathcal{T}} \rangle_{(1,1)}}{\min \langle m_{\mathcal{T}} \rangle_{(1,0)}} \approx 2\%,$$

$$\frac{\min \langle s_{\mathcal{T}} \rangle_{(1,0)} - \min \langle s_{\mathcal{T}} \rangle_{(1,1)}}{\min \langle s_{\mathcal{T}} \rangle_{(1,0)}} \approx 4\%.$$

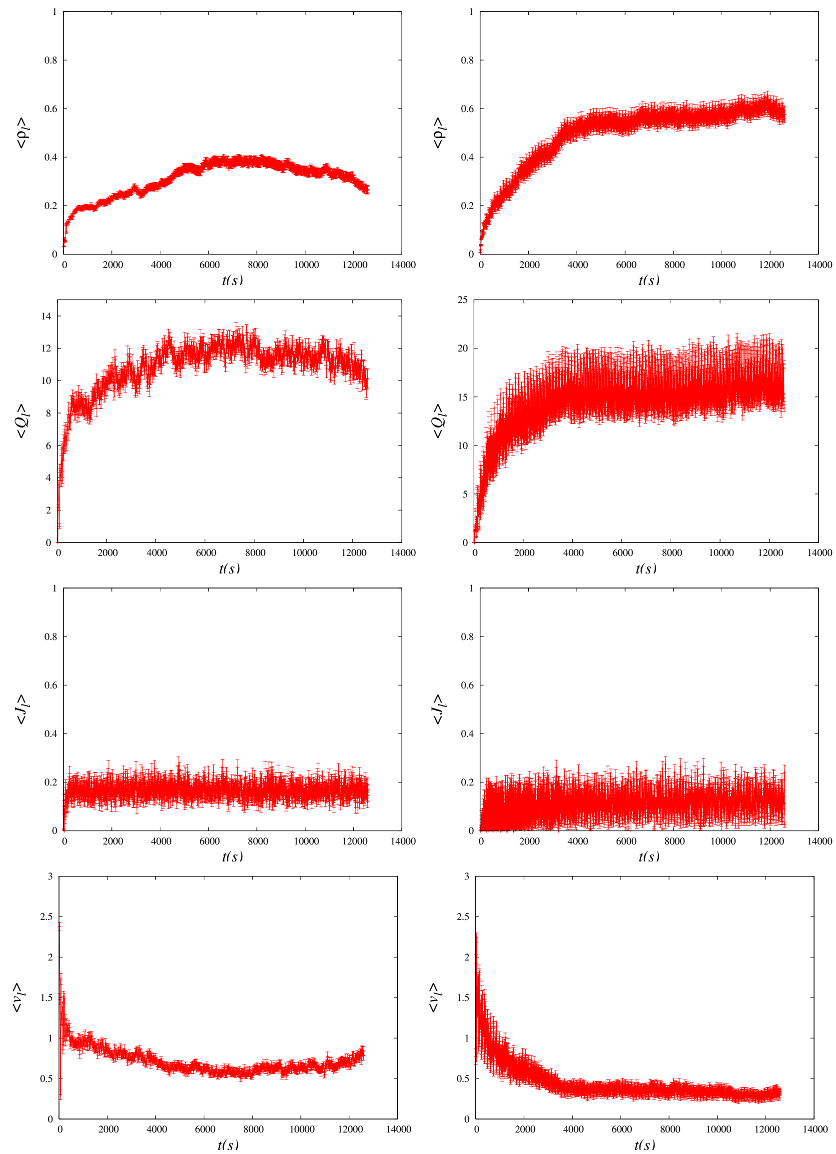


Figure 18. High Density. From top: SOTL (left) vs Fixed Cycle (right) evolution of the density, queue length, flow and space-mean speed, on a given bulk link, for the high density 4×4 square grid. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$.

6.4. Uniform low-density boundary conditions

To produce uniform low-density boundary conditions, for each boundary in-lane we set $\rho_{\lambda,\max} = 0.2$ and $\rho_{\lambda,\min} = 0.1$, and the turning probabilities were again chosen according to (8).

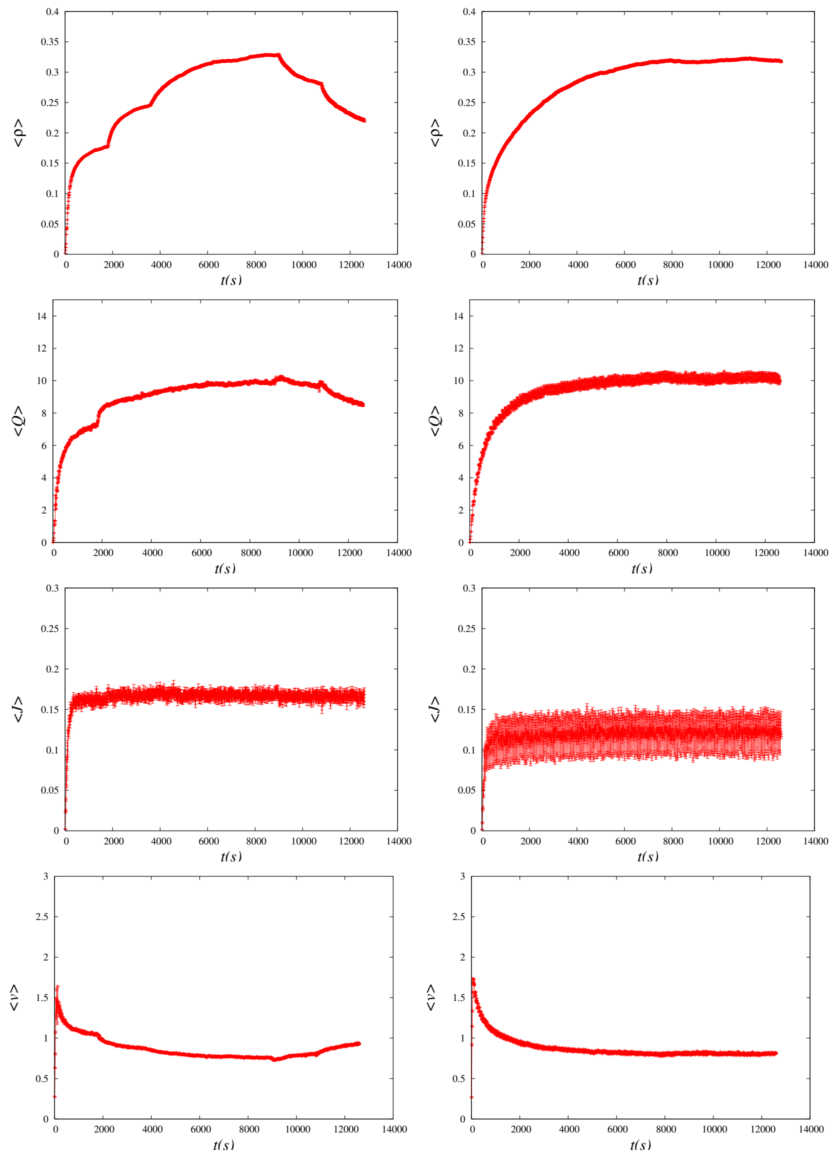


Figure 19. High Density. From top: SOTL (left) vs Fixed Cycle (right) evolution of the network-averaged density, queue length, and flow, for the high density 4×4 square grid. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$.

6.4.1. Comparing SOTL vs fixed-cycle traffic lights. In figure 21 we compare the evolution of the link observables ρ_l, Q_l, J_l and v_l for the low density square-lattice network for the adaptive SOTL update vs fixed-cycle traffic lights. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$. The fixed cycle times were again determined from the SOTL values midway through the morning peak hour. In figure 22 we compare the evolution of their network

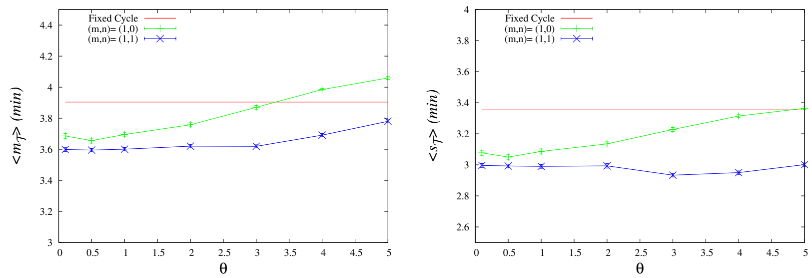


Figure 20. Mean travel time $\langle m_{\mathcal{T}} \rangle$ and its fluctuation $\langle s_{\mathcal{T}} \rangle$ vs SOTL threshold parameter θ, for the high-density 4×4 square grid, with the SOTL demand function (5) and SOTL demand exponents $(m, n) = (1, 0), (1, 1)$. The horizontal line shows the corresponding value for the system with fixed-cycle traffic lights.

averages. As for the Kew network studied in section 5, the means of both link and network observables are better for SOTL than for fixed-cycle traffic lights. Also, as before, the fluctuations for SOTL are significantly smaller.

6.4.2. Comparing upstream-only vs upstream-downstream SOTL. The average values of the travel time $m_{\mathcal{T}}$ and its fluctuation $s_{\mathcal{T}}$ are presented in figure 23. For $\theta < 2$, the $\langle m_{\mathcal{T}} \rangle$ curve is not very sensitive to the precise value of θ, while the $\langle s_{\mathcal{T}} \rangle$ curve has an optimal value at around $\theta \approx 2$. By contrast with the previous cases, there is no statistically significant difference between the $(1, 1)$ and $(1, 0)$ curves in this case, for either $\langle m_{\mathcal{T}} \rangle$ or $\langle s_{\mathcal{T}} \rangle$. See Table 4 for the exact numerical values. This is intuitively reasonable – for a network in which all links are freely flowing one would not expect an advantage from monitoring the downstream congestion, since it will always be negligible.

7. Conclusion

The main aims of this work have been to try and (partially) answer the questions of how adaptive signal strategies improve urban traffic flow, and what type of adaptive strategies perform best. To investigate these questions, we have developed a realistic

Table 3. Numerical values of the mean $\langle m_{\mathcal{T}} \rangle$ and fluctuation $\langle s_{\mathcal{T}} \rangle$ of the vehicle travel time for the high-density simulations of the $(1, 0)$ and $(1, 1)$ models. The statistical error shown corresponds to one standard deviation. The units are minutes. For comparison, the corresponding values using fixed-cycle traffic lights are $\langle m_{\mathcal{T}} \rangle_{\mathrm{fc}} = 3.90 \pm 0.01$ and $\langle s_{\mathcal{T}} \rangle_{\mathrm{fc}} = 3.35 \pm 0.01$.

	$(m, n) = (1, 0)$		$(m, n) = (1, 1)$	
θ	$\langle m_{\mathcal{T}} \rangle$	$\langle s_{\mathcal{T}} \rangle$	$\langle m_{\mathcal{T}} \rangle$	$\langle s_{\mathcal{T}} \rangle$
0.1	3.69 ± 0.02	3.08 ± 0.02	3.60 ± 0.01	3.00 ± 0.01
0.5	3.66 ± 0.02	3.05 ± 0.02	3.59 ± 0.01	2.99 ± 0.01
1.0	3.70 ± 0.01	3.09 ± 0.02	3.60 ± 0.01	2.99 ± 0.01
2.0	3.76 ± 0.01	3.13 ± 0.01	3.62 ± 0.01	2.99 ± 0.01
3.0	3.87 ± 0.01	3.23 ± 0.01	3.62 ± 0.01	2.93 ± 0.01
4.0	3.98 ± 0.01	3.31 ± 0.01	3.69 ± 0.01	2.95 ± 0.01
5.0	4.06 ± 0.01	3.36 ± 0.01	3.78 ± 0.01	3.00 ± 0.01

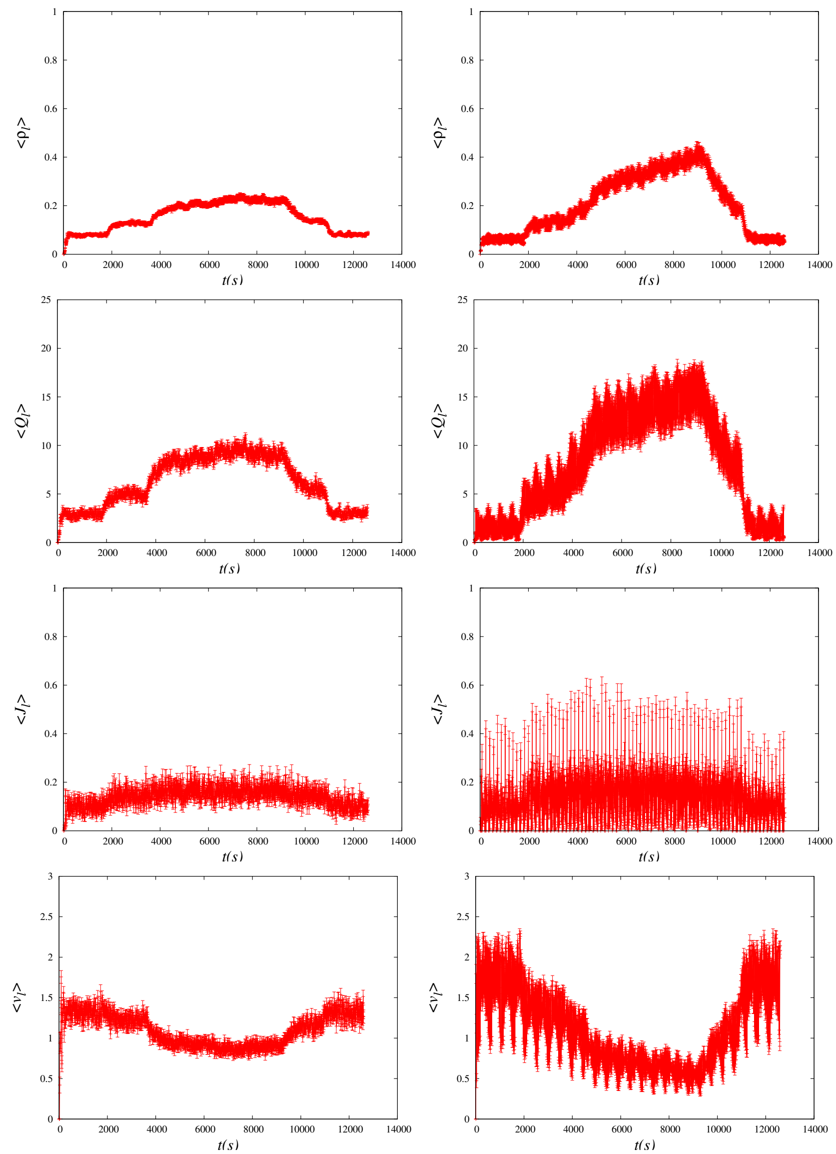


Figure 21. Low Density. From top: SOTL (left) vs Fixed Cycle (right) evolution of the density, queue length, flow and space-mean speed, on a given bulk link, for the low density 4×4 square grid. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$.

network traffic simulation model, which we used to simulate two different networks. The first is an existing road network in the Melbourne suburb of Kew, for which we have experimental data available as input into our simulation model. For comparison we have also simulated a square-lattice road network, in order to test network independent features and robustness.

On these two networks we have compared a non-adaptive signal system, with

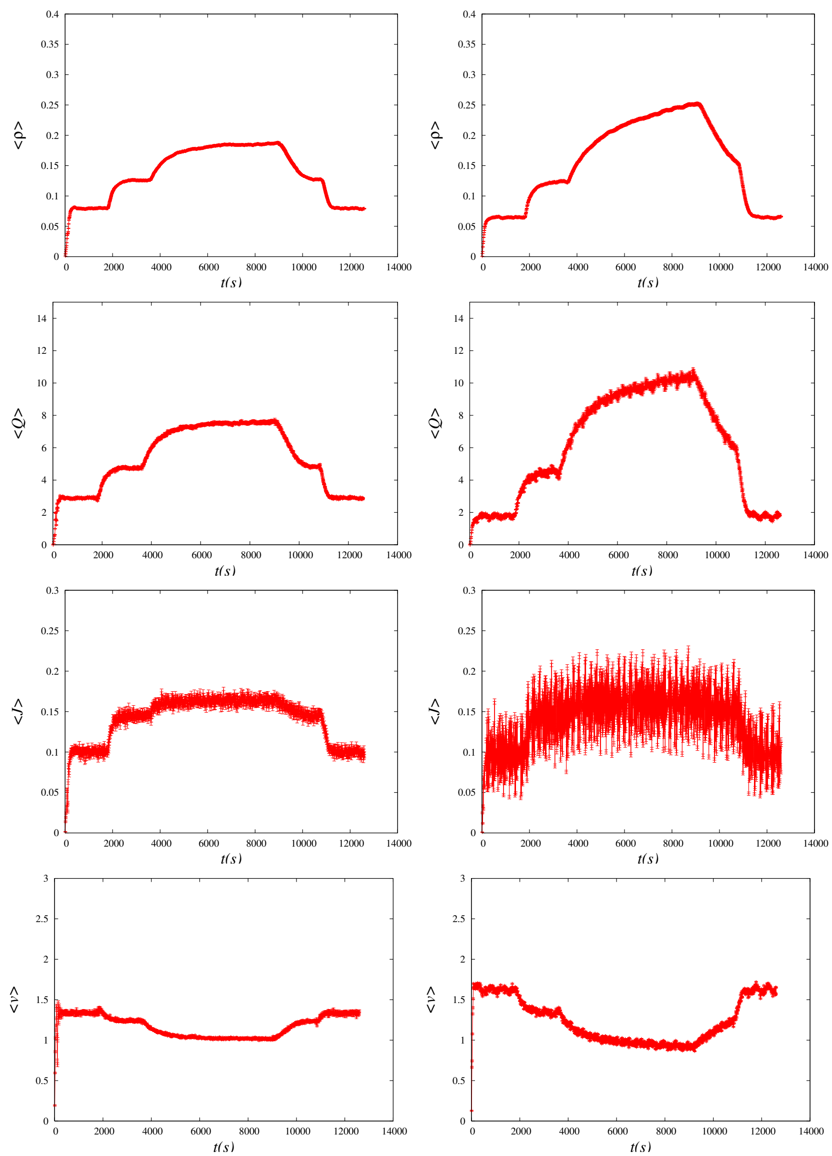


Figure 22. Low Density. From top: SOTL (left) vs Fixed Cycle (right) evolution of the network-averaged density, queue length, and flow, for the low density 4×4 square grid. The SOTL demand function (5) was used in the simulations, with SOTL demand exponents $(m, n) = (1, 1)$ and $\theta = 2$.

fixed-cycle traffic lights, with a version of the adaptive SOTL (Self Organizing Traffic Lights) introduced by Gershenson [26]. In the cases studied, we find that averages of observables such as travel time, density, flow, queue length and speed are almost always better for SOTL than for the fixed-cycle strategy. Moreover, the fluctuations in these observables are significantly smaller for SOTL. This suggests that a regular traffic signal system results in fairly large fluctuations in traffic observables compared

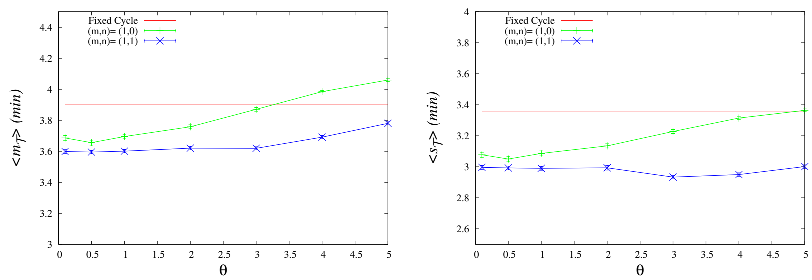


Figure 23. Mean travel time $\langle m_{\mathcal{T}} \rangle$ and its fluctuation $\langle s_{\mathcal{T}} \rangle$ vs SOTL threshold parameter θ, for the low-density 4×4 square grid, with the SOTL demand function (5) and SOTL demand exponents $(m, n) = (1, 0), (1, 1)$. The horizontal line shows the corresponding value for the system with fixed-cycle traffic lights.

to a deregulated self-organizing signal system. A similar observation was recently made by Lämmer and Helbing [27] in a self-organizing fluid-dynamic model for traffic flow in urban road networks.

On both networks we have also performed a comparison of two specific types of SOTL strategies; one which is informed only by the congestion on upstream links, and another which is informed by the congestion on both upstream and downstream links. Our results show that for four typical systems studied, provided the network is sufficiently congested the latter strategy is both more efficient (smaller travel times) and more reliable (smaller fluctuations in travel times) than the former. For an uncongested network we found that there was no discernible difference between the two strategies.

These results are only the tip of the ice berg. Firstly, it is of significant interest to obtain a more detailed understanding of how the relative efficiencies of the two SOTL strategies depends on network congestion. This is of crucial importance in determining whether the upstream-downstream strategy has any practical merit. Although in the systems we have studied, the efficiency gain from using the upstream-downstream strategy was modest, it is quite possible that there may be other regions of boundary input parameters in which the gains are far more significant. It is also conceivable that in some regimes of boundary input data the upstream-only strategy may in fact

Table 4. Numerical values of the mean $\langle m_{\mathcal{T}} \rangle$ and fluctuation $\langle s_{\mathcal{T}} \rangle$ of the vehicle travel time for the low-density simulations of the $(1, 0)$ and $(1, 1)$ models. The statistical error shown corresponds to one standard deviation. The units are minutes. For comparison, the corresponding values using fixed-cycle traffic lights are $\langle m_{\mathcal{T}} \rangle_{\text{fc}} = 2.53 \pm 0.01$ and $\langle s_{\mathcal{T}} \rangle_{\text{fc}} = 2.24 \pm 0.01$.

	$(m,n) = (1,0)$		$(m,n) = (1,1)$	
θ	$\langle m_{\mathcal{T}} \rangle$	$\langle s_{\mathcal{T}} \rangle$	$\langle m_{\mathcal{T}} \rangle$	$\langle s_{\mathcal{T}} \rangle$
0.1	2.18 ± 0.01	1.87 ± 0.01	2.16 ± 0.01	1.85 ± 0.01
0.5	2.17 ± 0.01	1.86 ± 0.01	2.16 ± 0.01	1.85 ± 0.01
1.0	2.21 ± 0.01	1.85 ± 0.01	2.17 ± 0.01	1.80 ± 0.01
2.0	2.22 ± 0.01	1.74 ± 0.01	2.23 ± 0.01	1.76 ± 0.01
3.0	2.39 ± 0.01	1.85 ± 0.01	2.41 ± 0.01	1.88 ± 0.01
4.0	2.54 ± 0.01	1.95 ± 0.01	2.54 ± 0.01	1.95 ± 0.01
5.0	2.68 ± 0.01	2.06 ± 0.01	2.67 ± 0.01	2.05 ± 0.01

be more efficient.

Furthermore, it is of great interest to study the effects of changing the network structure, in particular to study the above problems on much larger networks. To this end, the square-grid network discussed in section 6 is ideal – it is tailor made for studying the effect of increasing the network size parameters, L_x, L_y, while retaining the important features of a realistic network such as discussed in section 5. Preliminary simulations show that simulating such square-grid networks with 100 intersections is easily within reach computationally.

Acknowledgment

We thank the traffic engineers at VicRoads, in particular Adrian George and Andrew Wall, for many interesting and valuable discussions, as well as for making available to us the SCATS data for the Kew network. We further thank Ofer Biham, Kostya Borovkov, Ebrahim Fouladvand, Tony Guttmann, Reinout Quispel, Andreas Schadschneider, David Shteinman, Peter Taylor and Peter Van Der Kamp for discussions and support. This research was financially supported by the Australian Research Council.

Appendix A. Details of the network cellular automaton

Appendix A.1. Inflow.

For each lane, λ, of each boundary inlink we are given as input an inflow probability α_λ. At each instant of time, if the first cell of λ is empty, we add a new vehicle to this cell with probability α_λ. The value of α_λ will in general vary during the simulation, however in this section it suffices to think of α_λ as being fixed, since we are discussing how to implement the inflow at a given instant of time.

Since we will often estimate α_λ using an empirical stop-line occupancy, we typically model each boundary in-lane using a small number of cells. As a consequence of this, vehicles on boundary inlinks will most likely not have sufficient time to make *topological lane changes* (as described in section Appendix A.2). We therefore make the (quite reasonable) assumption that a vehicle in lane λ has decided to turn into a link which is connected to λ via one of the node's paths. Consequently, vehicles entering boundary in-lanes make their turning decisions according to the conditional probabilities $\mathbb{P}(l \to l'|\lambda)$, rather than $\mathbb{P}(l \to l')$.

Our procedure for inserting new vehicles into the network can now be summarized simply by algorithm 3.

Algorithm 3 (Inflow)

for *each boundary inlink l* ***do***
    for *each lane λ of l* ***do***
        if *the first cell of λ is vacant* ***then***
            With probability α_λ add a new vehicle with speed $v_{\max}$ to the first cell of λ
            Make a turning decision for the new vehicle using $\mathbb{P}(l \to l'|\lambda)$
        end if
    end for
end for

Finally, we need to express $\mathbb{P}(l \to l'|\lambda)$ in terms of the available data, $\mathbb{P}(l \to l')$. It is quite reasonable to assume that there is one unique lane $\lambda' \in l'$ for which a path $\lambda\lambda'$ from λ to l' exists. We therefore have

$$\mathbb{P}(l \to l'|\lambda) = \mathbb{P}(\lambda\lambda'|\lambda), \tag{A.1}$$

where $\mathbb{P}(\lambda\lambda'|\lambda)$ is the conditional probability that a vehicle on link l will traverse the path $\lambda\lambda'$, given that it is on lane λ.

Now, if $\mathbb{P}(\lambda\lambda'|l)$ is the probability that a given vehicle on link l will traverse the particular path $\lambda\lambda'$, then it is clear that

$$\mathbb{P}(\lambda\lambda'|\lambda) = \frac{\mathbb{P}(\lambda\lambda'|l)}{\sum_{\lambda\lambda''} \mathbb{P}(\lambda\lambda''|l)} \tag{A.2}$$

The sum in (A.2) is over all paths $\lambda\lambda''$ with in-lane λ. As an example, if we take $\lambda = \mathrm{in}(P_2)$ in figure 3 then the sum is over two paths $\lambda\lambda''$ where λ'' can be either $\lambda'' = \mathrm{out}(P_2)$ or $\lambda'' = \mathrm{out}(P_3)$. If there are $k_{ll'}$ paths $\lambda_i\lambda_i'$ connecting link l to link l', then clearly

$$\sum_{i=1}^{k_{ll'}} \mathbb{P}(\lambda_i\lambda_i'|l) = \mathbb{P}(l \to l'). \tag{A.3}$$

As an example, if we take $\lambda = \mathrm{in}(P_2)$ and $\lambda' = \mathrm{out}(P_2)$ in figure 3 then $k_{ll'} = 2$, whereas if we take $\lambda' = \mathrm{out}(P_3)$ we have $k_{ll'} = 1$. In fact, we shall assume that all possible paths are weighted equally

$$\mathbb{P}(\lambda_i\lambda_i'|l) = \frac{\mathbb{P}(l \to l')}{k_{ll'}}. \tag{A.4}$$

Combining (A.4), (A.2) and (A.1) then allows us to compute $\mathbb{P}(l \to l'|\lambda)$ from $\mathbb{P}(l \to l')$, as desired. Equation (A.4) seems a perfectly reasonable assumption, since a driver would be expected to care only about which link they were about to turn into, not which particular lane they use to do so.

Appendix A.2. Lane changing.

Lane changing in CA traffic models is a rather well-studied topic, and our implementation follows closely the ideas presented in [5, 30, 31, 32, 33]. We perform lane changing in two separate steps, so that we guarantee our network updates are carried out in parallel. For a given link, we firstly consider each occupied cell of each lane and *decide* which vehicles want to change lane, and all such vehicle's keep a record of which lane they want to change to. Then, once all vehicles have decided on their lane changes, we go through each cell of each lane again and *execute* the lane changes. This ensures that all vehicles make their lane changes based on information at the same time step. Once the lane changing decisions have been made, executing the lane changes is trivial, and so in this section we focus only on how to make the decisions. A vehicle may decide to change lanes for two distinct reasons:

(i) Topological: to ensure the vehicle can make its desired turn at the approaching intersection

(ii) Dynamic: to avoid bad traffic

At each time-step, we propose for each vehicle a specific lane change, and then decide whether or not it should be executed. We only propose lane changes from left-to-right on even time steps, and lane changes from right-to-left on odd time-steps. This ensures that we never have two vehicles competing for the same cell.

Appendix A.2.1. Topological lane changing. As already discussed, when a vehicle $\mathbf{v}$ first enters a link $l = mn$ we randomly choose one of the possible outlinks l' of the upcoming node n and assign l' to be the vehicle's *destination*. I.e. the vehicle has already decided to turn onto l' when it reaches n. This defines a set of *possible paths* for $\mathbf{v}$, denoted $\mathcal{P}_{\mathbf{v}} = \{P_i\}$, which is the subset of all the paths belonging to n which have inlink l and outlink l'. In order for $\mathbf{v}$ to make its desired turn at n it must traverse a path in $\mathcal{P}_{\mathbf{v}}$. It may be the case however that $\mathbf{v}$'s current lane λ is not the in-lane of any of the paths in $\mathcal{P}_{\mathbf{v}}$. In this case $\mathbf{v}$ will need to make one or more lane changes in order to enter a lane from which the desired turn is possible. In this context, we say a lane change $\lambda \mapsto \lambda'$ is *allowed* if the proposed new lane λ' is the in-lane of a path in $\mathcal{P}_{\mathbf{v}}$. In addition, we say a lane change is *needed* if λ is not the in-lane of a path in $\mathcal{P}_{\mathbf{v}}$, but λ' or a lane to the right (left) of λ' is the in-lane of a path in $\mathcal{P}_{\mathbf{v}}$, if λ' is to the right (left) of λ. *Allowed* lane changes are not necessarily *needed* because it may be the case that there already exists a $P \in \mathcal{P}_{\mathbf{v}}$ with $\mathrm{in}(P) = \lambda$. Conversely, a *needed* lane change may not be *allowed* according to this definition. Deciding if a proposed lane change is *allowed* and/or *needed* in the above senses of the terms is the only topological information required to decide whether to accept a proposed lane change. Algorithm 4 summarizes how to decide if a proposed lane change of vehicle $\mathbf{v}$ from lane λ to lane $\lambda' \sim \lambda$ is topologically *allowed* and/or *needed**.

Algorithm 4 (Topological lane changes)

Consider a vehicle $\mathbf{v}$ *on lane* λ*, and a proposed* $\lambda \mapsto \lambda'$
if *there exists* $P \in \mathcal{P}_{\mathbf{v}}$ *such that* $\mathrm{in}(P) = \lambda'$ ***then***
  $\lambda \mapsto \lambda'$ *is allowed*
else
  $\lambda \mapsto \lambda'$ *is not allowed*
end if
if *there exists* $P \in \mathcal{P}_{\mathbf{v}}$ *such that* $\mathrm{in}(P) = \lambda$ ***then***
  $\lambda \mapsto \lambda'$ *is not needed*
else
  if $\lambda' > \lambda$ *and there exists* $P \in \mathcal{P}_{\mathbf{v}}$ *such that* $\mathrm{in}(P) \geq \lambda'$ ***then***
    $\lambda \mapsto \lambda'$ *is needed*
  else if $\lambda' < \lambda$ *and there exists* $P \in \mathcal{P}_{\mathbf{v}}$ *such that* $\mathrm{in}(P) \leq \lambda'$ ***then***
    $\lambda \mapsto \lambda'$ *is needed*
  end if
end if

Appendix A.2.2. Dynamic lane changing. Suppose a vehicle $\mathbf{v}$ cannot reach free speed due to congestion in its current lane λ, and suppose further that the gap in $\lambda' \sim \lambda$ is larger than that in λ; see figure A1. This provides a dynamic incentive for $\mathbf{v}$ to change lanes $\lambda \mapsto \lambda'$, and when such an incentive exists we say $\lambda \mapsto \lambda'$ is *desirable*. We allow $\mathbf{v}$ to make such a lane change provided it is safe to do so, and provided $\lambda \mapsto \lambda'$ is topologically *allowed* (as defined above). We use algorithm 5 to decide whether $\lambda \mapsto \lambda'$ is *desirable* and/or *safe*. The definition of *safe* presented in

* By $\lambda \sim \lambda'$ we mean that lanes λ and λ' are adjacent; i.e. they are consecutive in the lane ordering of their link.

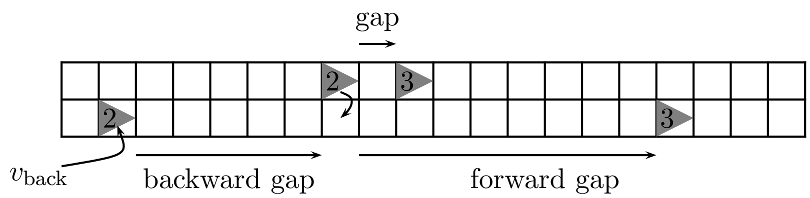


Figure A1. Typical situation arising in dynamic lane changing. Suppose $v_{\max} = 3$. We are proposing to move the vehicle of speed $v = 2$ on the left lane, to the right lane. Since $\min(v+1, \text{forwardgap}, v_{\max}) > \min(v+1, \text{gap}, v_{\max})$ the lane change is *desirable*. Since backwardgap $= 5 > v_{\text{back}}$ the lane change is also *safe*.

Algorithm 5 (Dynamic lane changes)

Consider a vehicle of speed v on lane λ, and a proposed $\lambda \mapsto \lambda'$
(See figure A1)
if backwardgap $> v_{\text{back}}$ ***then***
  $\lambda \mapsto \lambda'$ *is safe*
end if
if $\min(v+1, \text{forwardgap}, v_{\max}) > \min(v+1, \text{gap}, v_{\max})$ ***then***
  $\lambda \mapsto \lambda'$ *is desirable*
end if

---

algorithm 5 is stronger than merely ensuring vehicles avoid crashes; it ensures that the vehicle with speed v_{back} does not need to immediately decelerate.

Appendix A.2.3. Lane change decisions. At even (odd) time steps, we consider each link, and consider each lane of that link except the rightmost (leftmost) lane, and consider each cell on that lane which contains a vehicle. We then use algorithm 6 to decide whether or not that vehicle should perform a lane change to the lane to its right (left). Note that we accept *needed* but *unsafe* lane changes with a probability

---

Algorithm 6 (Lane change decision)

Consider a vehicle on cell i of a lane λ of length L, and a lane $\lambda' \sim \lambda$
if *cell i of lane λ' is unoccupied* ***then***
  if $\lambda \mapsto \lambda'$ *is needed* ***then***
    if $\lambda \mapsto \lambda'$ *is safe* ***then***
      Accept $\lambda \mapsto \lambda'$
    else if $\lambda \mapsto \lambda'$ *is not safe* ***then***
      Accept $\lambda \mapsto \lambda'$ with probability i/L
    end if
  else if $\lambda \mapsto \lambda'$ *is not needed but is allowed, desirable and safe* ***then***
    Accept $\lambda \mapsto \lambda'$ with probability p_{change}
  end if
end if

---

i/L, which increases as we proceed along the lane. This is a simple way to mimic the increasing urgency of getting into an appropriate lane to make a desired turn at the approaching intersection. We emphasize however, that even though we describe such lane changes as *unsafe*, they cannot cause a crash because we explicitly demand that cell i on lane λ' is empty. Perhaps a more accurate description for them would be *impolite* lane changes, since their effect is to force other vehicles to decelerate. We also remark that in practice we set $p_{\text{change}} < 1$ to avoid platoons oscillating back and forth on consecutive time-steps.

Appendix A.3. NaSch Dynamics.

Nagel-Schreckenberg(NaSch) dynamics refers to a standard one-dimensional stochastic dynamics, which is routinely utilized in freeway models. Each lane is divided up into cells of length $7.5m$, which represents the approximate space occupied by a vehicle in a jam. We assume each time-step corresponds to 1 second, so that a vehicle may only have one of a discrete set of speeds which are multiples of $27km/h$. The key step in performing the NaSch updates is to compute new velocities for each vehicle. Suppose at time t a vehicle with speed $v_t \in \{0, 1, \ldots, v_{\max}\}$ is located in cell x_t, and has *headway* (number of empty cells ahead) equal to h_t. Then the maximum speed this vehicle can safely achieve at the next time step is taken to be $v_{\text{safe}} = \min(v_t + 1, v_{\max}, h_t)$, which allows for unit acceleration provided the speed limit is obeyed and crashes are avoided. Provided $v_{\text{safe}} > 0$, a random unit deceleration is then applied so with probability p_{noise} the new speed is $v_{t+1} = v_{\text{safe}} - 1$, otherwise $v_{t+1} = v_{\text{safe}}$. Finally, in the bulk of the lane, the vehicle hops v_{t+1} cells ahead, so that $x_{t+1} = x_t + v_{t+1}$. For each lane, all vehicles in the bulk of the lane are updated in this way in parallel. It is known from freeway studies that the random deceleration step is crucial for obtaining a realistic model [1]. In our simulations we set $p_{\text{noise}} = 0.2$ if $v < v_{\max}$ and $p_{\text{noise}} = 0.5$ if $v = v_{\max}$, and so we have what is known as a velocity dependent randomization (VDR) model in the statistical mechanics literature (see e.g. [23]). If a vehicle $\mathbf{v}$ lies in cell $x_t \geq L - v_{\max} - 1$ and has no occupied cells in front of it then we set $h_t = v_{\max}$; if $\mathbf{v}$ has sufficiently low speed it will then be updated via NaSch in the same way as any other vehicle on the lane, otherwise such vehicles are handled separately by the mark paths and clear paths routines, see Appendix A.4 and Appendix A.5.

Algorithm 7, which uses the *NaSch* speed function, is applied to each lane of each bulk link, at each time step. The association of vehicles with paths, $\mathbf{v} \leftrightarrow P$, and the *marking* of vehicles as *needing to stop*, referred to in algorithm 7, is performed by the *mark paths* routine; see algorithm 8 in section Appendix A.4. We emphasize here however that it is only the very last vehicle on a lane that may be associated with paths or required to stop.

Appendix A.4. Mark Paths.

By *marking paths* for a given link $l = mn$ we mean that we consider each lane λ of l, and determine whether or not λ has a vehicle $\mathbf{v}$ which is sufficiently close to the end of λ, and traveling sufficiently fast, that a naive application of NaSch dynamics could move $\mathbf{v}$ past the end of λ♯. If this is the case we search for a path P of the active phase $\mathcal{P}_{\text{active}}$ which has $\text{in}(P) = \lambda$ and $\text{out}(P) \in \text{turn}(\mathbf{v})$, where $\text{turn}(\mathbf{v})$ denotes the

♯ We emphasize that due to the nature of NaSch dynamics, there can be at most one such vehicle at each time step.

Algorithm 7 (NaSch – network)

Consider lane λ *of length* L
for *every cell* $i = 0, \ldots, L-1$ ***do***
  if *cell(i) contains a vehicle* $\mathbf{v}$ ***then***
    if $\mathbf{v} \leftrightarrow P$ *for some path* P ***then***
      return
    end if
    if $\mathbf{v}$ *has been marked as needing to stop at the end of* λ ***then***
      Stop $\mathbf{v}$ *at the end of* λ
      return
    end if
    Set speed$(\mathbf{v}) = NaSch(\mathbf{v})$
    Move $\mathbf{v}$ *from* i *to* $i + NaSch(\mathbf{v})$
  end if
end for

unique outlink of n onto which $\mathbf{v}$ originally decided to turn when it first entered link l. If there exists such a path P then $\mathbf{v}$ could make its desired turn during the current iteration by traversing P, and so in such a case we associate P and $\mathbf{v}$, a relationship that we abbreviate with $P \leftrightarrow \mathbf{v}$. When a path has been associated with a vehicle in this way we say it has been *marked*. The actual traversal of $\mathbf{v}$ along P will take place when we *clear paths*, provided there are no other marked paths to which P must give way; this is discussed further in section Appendix A.5.

In practice, we perform path marking by applying algorithm 8 to each lane of each link of the network. Recall that $\mathcal{P}_n$ is the set of all paths of node n and that

$$\mathcal{P}_{\mathbf{v}} = \{P \in \mathcal{P}_n : \mathrm{in}(P) \in \mathrm{link}(\mathbf{v})\,\&\,\mathrm{out}(P) \in \mathrm{turn}(\mathbf{v})\}.$$

Some comments are in order. Firstly, note that we compute the NaSch speed using $p_{\mathrm{noise}} = 0$, regardless of the value we use in the NaSch updates, to ensure that we identify all vehicles that *could* possibly move past the end of their lane in one NaSch update. Secondly, note that the set $\mathcal{A}_\lambda$ is the set of all paths P which are available for $\mathbf{v}$ to traverse during the current iteration, without regard to whether they are consistent with $\mathbf{v}$'s turn decision; i.e. regardless of whether or not they satisfy $\mathrm{out}(P) \in \mathrm{turn}(\mathbf{v})$. Therefore, $\mathcal{A}_\lambda \cap \mathcal{P}_{\mathbf{v}}$ is simply the set of all $P \in \mathcal{A}_\lambda$ for which $\mathrm{out}(P) \in \mathrm{turn}(\mathbf{v})$. If there are any paths at all in $\mathcal{P}_n$ along which a vehicle on lane λ can move to the link $\mathrm{turn}(\mathbf{v})$, then we demand that $\mathbf{v}$ may only be associated with such a path, even if no such paths belong to the current $\mathcal{A}_\lambda$. If such paths do indeed exist but do not belong to the current $\mathcal{A}_\lambda$ then $\mathbf{v}$ is flagged as needing to stop at the end of λ. The vehicle will then wait at the lights until an appropriate phase, consistent with its turn decision, becomes active. However, it is possible that despite all the topological lane changing, a vehicle $\mathbf{v}$ may end up in a lane which is inconsistent with its desired turn decision††. When such a case arises, rather than let $\mathbf{v}$ block traffic we demand that it give up on its turn decision and simply randomly chose one of the paths currently available to it if one exists, otherwise we again stop $\mathbf{v}$ at the end of λ. Recall that if $\mathbf{v}$ is flagged

††Empirically, for the Kew network this seems to happen to about 3% of vehicles, which therefore does not significantly affect the effective origin-destination data encoded in the turning probabilities.

Algorithm 8 (Mark paths)

Suppose the last occupied cell of lane λ *of link* $l = mn$ *contains vehicle* $\mathbf{v}$
Let $\mathcal{A}_\lambda = \{P \in \mathcal{P}_{\text{active}} : \text{in}(P) = \lambda \ \&\ \text{out}(P) \text{ has space}\}$
if $cell(\mathbf{v}) + NaSch_{p_{\text{noise}}=0}(\mathbf{v}) \geq length(\lambda)$ ***then***
  if $\{P \in \mathcal{P}_{\mathbf{v}} : \text{in}(P) = \lambda\} \neq \emptyset$ ***then***
    if $\mathcal{A}_\lambda \cap \mathcal{P}_{\mathbf{v}} \neq \emptyset$ ***then***
      UAR, choose $P \in \mathcal{A}_\lambda \cap \mathcal{P}_{\mathbf{v}}$ *and associate* $P \leftrightarrow \mathbf{v}$
    else
      Mark $\mathbf{v}$ *as needing to stop at the end of* λ
    end if
  else
    if $\mathcal{A}_\lambda \neq \emptyset$ ***then***
      UAR, choose $P \in \mathcal{A}_\lambda$ *and associate* $P \leftrightarrow \mathbf{v}$
    else
      Mark $\mathbf{v}$ *as needing to stop at the end of* λ
    end if
  end if
end if

as having to stop at the end of λ then this move is actually performed by NaSch; see Appendix A.3. Finally, in algorithm 8 we use the prescription in algorithm 9 to determine if a lane *has space*.

Algorithm 9 (Has space)

Consider path P
if $\text{out}(P)$ *belongs to a bulk link* ***then***
  if *the first cell of* $\text{out}(P)$ *is empty* ***then***
    $\text{out}(P)$ *has space*
  end if
else if $\text{out}(P)$ *belongs to a boundary link* ***then***
  $\text{out}(P)$ *has space with probability* $(1 - \rho_{\lambda,1})$
end if

Finally, note that we perform path marking *before* the NaSch updates because in order for algorithm 1 to be parallel we need the determinations of whether a given lane has an empty first cell to occur *before* we update these cells. We also require the marking information within NaSch so that we can correctly stop vehicles on the end of their lane if need be.

Appendix A.5. Clear paths.

Recall that for a given node, and a given phase, each path has associated with it a list (possibly empty) of other paths in the same phase to which it must give way. If path P' is listed in path P's give-way list, and both P and P' are marked, then P will not be cleared during the current iteration. In this sense P' has *priority* over P. In practice, P' might represent a vehicle traveling straight through a four-way intersection while P represents a vehicle traveling in the opposite direction and wishing to turn right

(cf. paths P_6 and P_3 in figure A1). Algorithm 10 describes the clear path routine in detail.

Algorithm 10 (Clear paths)

Consider node n
for *each marked path* $P \in \mathcal{P}_n$ ***do***
  if *there is another marked path to which* P *must give way* ***then***
    Move the vehicle $\mathbf{v} \leftrightarrow P$ *to the last cell of* $\mathrm{in}(P)$
    Give $\mathbf{v}$ *speed 0*
    Disassociate P *and* $\mathbf{v}$
  else
    if $\mathrm{out}(P)$ *belongs to a bulk link* ***then***
      Move the vehicle $\mathbf{v} \leftrightarrow P$ *to the first cell of* $\mathrm{out}(P)$
      if $\mathit{speed}(\mathbf{v}) = 0$ ***then***
        Set $\mathit{speed}(\mathbf{v}) = 1$
      end if
    else if $\mathrm{out}(P)$ *belongs to a boundary link* ***then***
      Delete $\mathbf{v} \leftrightarrow P$
    end if
  end if
end for

Appendix A.6. Choose phases

This depends on the choice of signal rules, of which there are infinitely many one may consider. Perhaps the simplest rules are simply fixed cycle rules; for each node we have an ordered list of phases $(\mathcal{P}_1, \ldots, \mathcal{P}_m)$ and an ordered list of *split* times $(t_1, \ldots, t_m)$. We then cycle through these phases according to the corresponding split times; phase $\mathcal{P}_i$ is the active phase for t_i iterations, then $\mathcal{P}_{i+1}$ is the active phase for t_{i+1} iterations, etc.

More sophisticated rules may choose the active phase based on the actual network configuration. One examples is the self-organized traffic lights discussed in section 3. The implementation of this rule is given in Algorithm 2.

References

[1] Chowdhury D, Santen L and Schadschneider A 2000 *Phys. Rep.* **329** 199–329
[2] Helbing D 2001 *Rev. Modern Phys.* **73** 1067–1141
[3] Schadschneider A 2002 *Physica A* **313** 153–187
[4] Nagel K, Wagner P and Woesler R 2003 *Oper. Res.* **51** 681–710
[5] Nagel K 2004 *Multi-agent transportation simulation* URL `https://svn.vsp.tu-berlin.de/repos/public-svn/publications/kn-old/book/book.pdf`
[6] Maerivoet S and De Moor B 2005 *Physics Reports* **419** 1–64
[7] Nagel K and Schreckenberg M 1992 *Journal de Physique* **2** 2221–2229
[8] Biham O, Middleton A A and Levine D 1992 *Phys. Rev. A* **46** R6124
[9] Chopard B, Luthi P O and Queloz P A 1996 *J. Phys. A: Math. Gen.* **29** 2325–2336
[10] Chowdhury D and Schadschneider A 1999 *Phys. Rev. E* **59** 1311–1314
[11] Brockfeld E, Barlovic R, Schadschneider A and Schreckenberg M 2001 *Phys. Rev. E* **64** 056132

[12] Barlović R, Brockfeld E, Schadschneider A and Schreckenberg M 2003 *Traffic and Granular Flow 01* ed Fukui M, Sugiyama Y, Schreckenberg M and Wolf D E
[13] Esser J and Schreckenberg M 1997 *Internat. J. Modern Phys. C* **8** 1025–1036
[14] Schreckenberg M, Neubert L and Wahle J 2001 *Future Generation Computer Systems* **17** 649–657
[15] Cetin N, Nagel K, Raney B and Voellmy A 2002 *Comput. Phys. Comm.* **147** 559–564
[16] Scellato S, Fortuna L, Frasca M, Gómez-Gardeñes J, and Latora V 2010 *Eur. Phys. J. B* **73** 303–308
[17] D Helbing J S and Lämmer S 2007 *Networks and Heterogeneous Media* **2** 193–210
[18] Faieta B and Huberman B A 1993 Firefly: A Synchronization Strategy for Urban Traffic Control URL `http://handle.dtic.mil/100.2/ADA270872`
[19] Fouladvand M E and Nematollahi M 2001 *Eur. Phys. J. B* **22** 395–401
[20] Huang D and Huang W 2003 *Phys. Rev. E* **67** 056124
[21] Sasaki M and Nagatani T 2003 *Physica A* **325** 531–546
[22] Sekiyama K, Nakanishi J, Takagawa I, Higashi T and Fukuda T 2001 *IEEE Int. Conf. Syst. Man. Cybern.* **4** 24812486
[23] Barlovic R, Huisinga T, Schadschneider A and Schreckenberg M 2002 *Phys. Rev. E* **66** 046113
[24] Fouladvand M E, Shaebani M R and Sadjadi Z 2004 *J. Phys. Society Japan* **73** 3209
[25] Lämmer S 2007 *Reglerentwurf zur dezentralen Online-Steuerung von Lichtsignalanlagen in Straennetzwerken (Controller design for a decentralized control of traffic lights in urban road networks). Ph.D. thesis. Dresden University of Technology*
[26] Gershenson C 2005 *Complex Systems* **16** 29–53
[27] Lämmer S and Helbing D 2008 *J. Stat. Mech. Theory Exp.* **2008** 04019
[28] Cools S B, Gershenson C and D'Hooghe B 2008 *Advances in Applied Self-organizing Systems* (New York: Springer) chap 3
[29] Gershenson C and Rosenblueth D 2009 *arXiv:0907.1925v1*
[30] Rickert M, Nagel K, Schreckenberg M and Latour A 1996 *Physica A* **231** 534–550
[31] Wagner P, Nagel K and Wolf D E 1997 *Physica A* **234** 687–698
[32] Nagel K, Wolf D E, Wagner P and Simon P 1998 *Phys. Rev. E* **58** 1425–1437
[33] Knospe W, Santen L, Schadschneider A and Schreckenberg M 1999 *Physica A* **265** 614–633